Nagst de populaire serie zakboekjes die reeds zijn verschenen voor een aantal microcomputers, is dit (na de 6502) het tweede zakboekje voor een microprocessor.
De Z80 wordt toegepast in een groot aantal zeer bekende microcomputers, o.a. Sinclair ZX81, ZX Spectrum(+), in de reeds op de markt gebrachte en nog te verschijnen MSX-computers (Philips, Sony Hit-Bit, Spectravideo, Panasonic, Goldstar enz.).
Onder meer worden behandeld de interne architectuur, adresseringsmogelijkheden en de verschillende registers.
Ten slotte wordt een alfabetisch overzicht gegeven van de volledige instructieset van deze processor.

ISBN 90 201 1808 0

j.b. vonk – zakboekje Z80

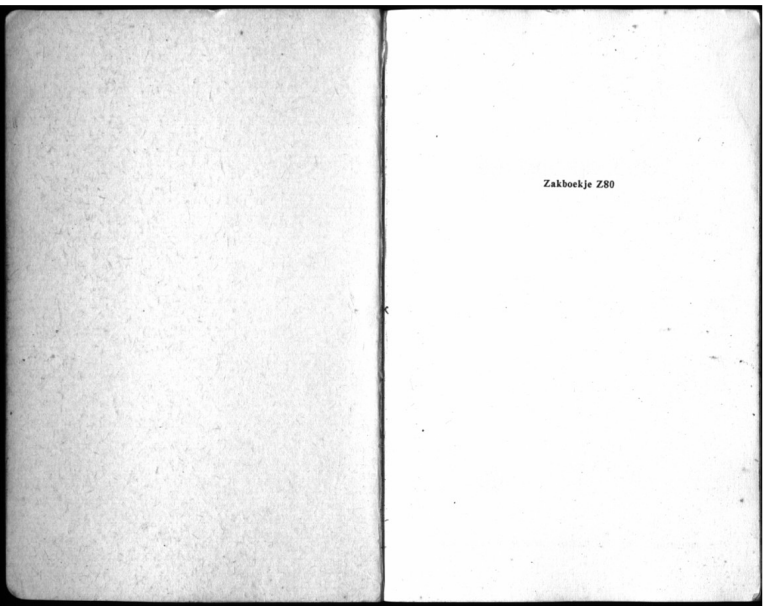
zakboekje

Z-80

architectuur
adresseringswijzen
instructieset

j.b. vonk

kluwer technische boeken


Zakboekje Z80

Zakboekje Z80

architectuur
adresseermodes
instructieset

J. B. Vonk

Kluwer Technische Boeken B.V. Deventer - Antwerpen

Lay-out: W.Wijnolts

ISBN 90 201 1808 0
D/1985/0108/197

1e druk 1985

Woord vooraf

Er is al veel over de Z80 geschreven. Ik heb dan ook niet de pretentie met dit zakboekje opzienbarend nieuws te brengen. Wel heb ik geprobeerd veel feiten te rangschikken tot een overzichtelijk geheel. Daarbij stond me het volgende gebruik voor ogen:

- het programmeren van een computer met behulp van een assembler of direct in machinetaal;
- het werken met een single board computer;
- het bouwen van schakelingen.

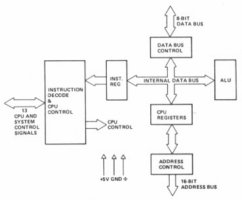
Hier en daar zijn onderwerpen iets uitgebreider behandeld dan misschien van een zakboekje mag worden verwacht. Het betreft dan zaken die in Z80-lectuur of in algemene boeken over microprocessors slechts summier aan bod komen.

De firma Zilog bedank ik hartelijk voor het afstaan van illustratiemateriaal.

J. B. Vonk

Inhoud

1. Interne architectuur



Afb. 1.1. Interne architectuur.

Afbeelding 1.1 toont in een blokdiagram het inwendige van de Z80. Het is een schematische voorstelling en staat derhalve los van de fysieke opbouw van de chip. De functie van de diverse elementen wordt in het nu volgende globaal uiteengezet.

DATABUS CONTROL
Regelt het verkeer over de interne databus. Bij de instructie LD H,B (laadt register H met de inhoud van register B) bijvoorbeeld wordt de inhoud van register B over de interne databus gekopieerd in een tijdelijk register dat bij de ALU hoort. De inhoud van het tijdelijke register wordt daarna via de databus gekopieerd in register H. Bij het ophalen van een instructie

gaat de van het geheugen komende data via de interne databus naar het instructieregister.

INSTRUCTIEREGISTER
Bevat de laatst opgehaalde instructie.

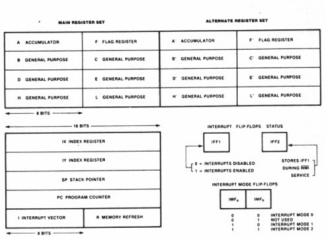
INSTRUCTIEDECODER & CPU-CONTROL
Hierin wordt de instructie gedecodeerd waarna alle signalen worden opgewekt die binnen en buiten de Z80 nodig zijn om de instructie uit te voeren.

ADDRESS CONTROL
Regelt het verkeer over de interne adresbus.

ALU
Arithmetic & Logic Unit (rekenkundige en logische eenheid). Hierin vindt de uitvoering plaats van de rekenkundige en logische instructies (optellen, aftrekken, increment en decrement, vergelijken, logische AND, OR en XOR) en de instructies voor schuiven en roteren, bit test, set, reset en vergelijken.

CPU-REGISTERS
Algemene registers
De registers A, B, C, D, E, H en L zijn door de programmeur te gebruiken 8-bit registers. De registers B en C, D en E, H en L kunnen als registerpaar worden gebruikt en vormen dan de 16-bit registers BC, DE en HL. Hiermee kunnen 16-bit berekeningen worden gemaakt, waarbij HL als accumulator fungeert, maar ook kan de inhoud van een 16-bit register als adrespointer worden gebruikt.
Register A is de accumulator. Bij alle 8-bit rekenkundige en logische bewerkingen, behalve increment en decrement, bevindt zich hierin één van de operanden en, na de bewerking, het resultaat.
Vlagregister F (flags) is een 8-bit register waarvan er zes door de Z80 als vlag worden gebruikt. Hoofdstuk 6 is geheel aan het vlagregister gewijd.
Alle bovengenoemde registers zijn in tweevoud uitgevoerd. Met exchange-instructies kunnen de inhouden van de alternatieve, gelijknamige registers worden verwisseld. De alternatieve registerset kan worden gebruikt als kladgeheugen bij berekeningen maar ook om de inhoud van registers te redden bij een

interrupt. Dit laatste is natuurlijk niet mogelijk als interrupts genest kunnen voorkomen.



Afb. 1.2. Overzicht van de registers.

Index-registers
De twee index-registers IX en IY worden gebruikt voor geïndexeerde adressering. De inhoud van de registers is een 16-bit adres waarbij een in de instructie opgegeven verplaatsing wordt opgeteld. Het zo ontstane adres wordt op de adresbus geplaatst. De verplaatsing is een getal van 1 byte in twee-complementsnotatie en kan dus zowel positief als negatief zijn.

Program counter (programmateller PC)
De PC verzorgt het sequentiële verloop van een programma. Het is een 16-bit register waarvan de inhoud wijst naar de geheugenlocatie waarin zich een op te halen instructie of data bevindt. Is de instructie of de data opgehaald dan wordt de inhoud van de PC automatisch met één verhoogd en wijst dan naar de volgende instructie of op te halen data. Een reset zet de PC op nul. Bij absolute sprongen of calls wordt de inhoud van de PC vervangen door een nieuw adres. Bij relatieve sprongen wordt bij de inhoud

van de PC een in de instructie opgegeven verplaatsing opgeteld. De verplaatsing is een getal van 1 byte in twee-complementsnotatie en kan dus zowel positief als negatief zijn.

Stack pointer (stapelaanwijzer SP)
De SP is een 16-bit register waarin het adres van de laatst bezette plaats van de stapel staat. De stapel kan overal in RAM-geheugen staan. Om het begin van de stapel te bepalen kan SP op verschillende manieren worden geladen: register geadresseerd, uitgebreid en onmiddellijk uitgebreid geadresseerd. De stapel werkt van boven naar beneden. Een PUSH-instructie zet de inhoud van een 16-bit register of registerpaar op de stapel en vermindert de inhoud van SP met twee. Een POP-instructie zet de twee laatst aan de stapel toegevoegde bytes in een 16-bit register of registerpaar en vermeerdert de inhoud van SP met twee.
De stapel is dus 'last in first out' (lifo) georganiseerd. Wat er het laatst door een PUSH-instructie is opgezet, wordt er het eerst door een POP-instructie afgehaald.
De stapel kan worden gebruikt om bijvoorbeeld tussenresultaten van een berekening tijdelijk weg te zetten, voor het doorgeven van parameters aan een subroutine of om de inhoud van registerparen te redden bij een interrupt. CALL- en RST-(restart)-instructies zetten de inhoud van de PC op de stapel alvorens de PC met het beginadres van de subroutine te laden. Een return-instructie 'POPt' de top van de stapel terug in de PC. De inhoud van SP kan programmatisch worden bepaald.

Interruptvector-register I
Werkt de Z80 in interrupt-mode 2 dan moet het randapparaat dat een maskeerbare interrupt aanvraagt een 8-bit vector op de databus zetten. Samen met de inhoud van het interruptvector-register I vormt de 8-bit vector een 16-bit adres waarnaar een CALL wordt uitgevoerd.
De interrupts worden uitgebreid behandeld in hoofdstuk 7.

Memory-refresh-register R
Tijdens het decoderen van een instructie, dus na elke opcode fetch, zet de Z80 de inhoud van de registers I en R op de adresbus en voert een refresh uit voor dynamische RAM. Na iedere afzonderlijke verplaatsing of vergelijking bij blokverplaatsing, vergelijken of input/output-instructies voert de Z80 twee

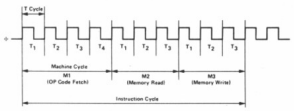
refreshes uit. Alleen de laagste helft van de adresbus, A0 t/m A7, waarop de inhoud van R staat, is van belang. De inhoud van R, die na elke refresh met één wordt verhoogd, vormt het rij- of kolomadres voor de dynamische RAM. De automatische verhoging met één betreft alleen de laatste zeven bits van R. Het hoogste bit blijft constant. R kan worden geladen met LD R,A en worden bekeken via LD A,R.

Interrupt-flipflops
De toestand van de interrupt-flipflops IFF1 en IFF2 bepaalt of maskeerbare interrupts al dan niet zijn toegestaan. Maskeerbare interrupts zijn toegestaan als de flipflops zijn geset. Het setten gebeurt door de instructie EI (enable interrupts). Zijn de flipflops gereset dan worden maskeerbare interrupts niet toegestaan. Het resetten gebeurt door de instructie DI (disable interrupts).
De inhoud van de interruptmode-flipflops bepaalt de respons van de Z80 op een maskeerbare interrupt. Uit de drie manieren waarop de Z80 op een maskeerbare interrupt kan reageren wordt gekozen met de instructies IM 0, IM 1 en IM 2 (interrupt mode). Deze instructies geven de interruptmode-flipflops hun waarde.
De interrupts worden uitgebreid behandeld in hoofdstuk 7.

BUSSEN
De verbindingen met de andere delen van het systeem, de databus, de adresbus en controlbus, worden behandeld in hoofdstuk 9.

2. Machinecyclus en T-cyclus

De uitvoering van elke instructie bestaat uit het opeenvolgend doorlopen van één tot zes basisoperaties ofwel machinecycli. Het uitvoeren van een machinecyclus neemt een aantal klokperioden of T-cycli in beslag. Het aantal machinecycli komt overeen met het aantal malen dat de Z80 geheugen of I/O aanspreekt voor het ophalen en uitvoeren van een instructie.
De eerste machinecyclus van een instructie, tijdens welke de opcode wordt opgehaald, de M1-cyclus, duurt 4, 5 of 6 T-cycli. Daarop volgende machinecycli duren 3, 4 of 5 T-cycli. Hierbij moeten nog eventuele wait states worden opgeteld. (Een wait state is een klokperiode tijdens welke de Z80 wacht tot geheugen of I/O klaar is. Meerdere wait states kunnen opeenvolgend voorkomen.)



Afb. 2.1. Instructie bestaande uit drie machinecycli en tien T-cycli.

In de instructieset in hoofdstuk 8 is van iedere instructie aangegeven hoeveel machinecycli en T-cycli deze in beslag neemt. Aan de hand hiervan kan de uitvoeringstijd van een instructie als volgt worden berekend.
Als f de frequentie van de klok is duurt een periodetijd of T-

cyclus 1/f sec. Dus bij een klokfrequentie van 4 MHz:

periodetijd of T-cyclus $= \frac{1}{4.10^6} = 2{,}5.10^{-7} = 250$ nanosec.

Een instructie waarvan de uitvoering 4 T-cycli in beslag neemt duurt dus: $4.250.10^{-9} = 1$ microseconde.
In onderstaande tabel volgt de lengte van een T-cyclus voor de diverse Z80-typen bij gebruik van de maximaal toelaatbare frequentie.

Z80	2,5 MHz	400 ns	
Z80A	4 MHz	250 ns	
Z80B	6 MHz	167 ns	
Z80H	8 MHz	125 ns	
Z80L	1 MHz	1000 ns	(low power-versies)
Z80L	2,5 MHz	400 ns	

3. Instructieformaten

De instructies van de Z80 kunnen worden ingedeeld naar formaat, dat wil zeggen naar het aantal bytes dat ze in beslag nemen. Het instructieformaat van de Z80 is één, twee, drie of vier bytes. Binnen de instructie zelf valt het volgende te onderscheiden:

De operatiecode of opcode
Deze bepaalt de handeling die moet worden uitgevoerd. De Z80 kent instructies met één-, twee- of drie-bytes opcode. Één- of twee-bytes opcode kunnen worden gevolgd door zowel één- als twee-bytes data. De betekenis van de data hangt af van de opcode. Het kan gaan om een getal dat in een register of een registerpaar moet worden geladen, om de verplaatsing voor een relatieve sprong, een 16-bit adres enz.
Bij een aantal instructies volgt op een twee-byte opcode een databyte en daarna één byte opcode.

DE ÉÉN-BYTE INSTRUCTIE
Deze bestaat alleen uit een opcode, eventueel voorzien van één of twee velden om interne registers aan te wijzen.

Voorbeelden:

LD r,(HL)	0 1 < r > 1 1 0	(in r registercode)
DI	1 1 1 1 0 0 1 1	

DE TWEE-BYTE INSTRUCTIE
Bestaat uit twee bytes opcode of één opcode-byte en één data-byte.
Voorbeelden:

RETI	1 1 1 0 1 1 0 1	opcode
	0 1 0 0 1 1 0 1	opcode

ADD A,7	1 1 0 0 0 1 1 0	opcode
	0 0 0 0 0 1 1 1	data (7)

DE DRIE-BYTE INSTRUCTIE
Bestaat uit één byte opcode en twee databytes of twee bytes opcode en één databyte.

Voorbeelden:

JP FF00	1 1 0 0 0 0 1 1	opcode
	0 0 0 0 0 0 0 0	data (laag deel adres)
	1 1 1 1 1 1 1 1	data (hoog deel adres)
INC (IY+0F)	1 1 1 1 1 1 0 1	opcode
	0 0 1 1 0 1 0 0	opcode
	0 0 0 0 1 1 1 1	data (verplaatsing)

DE VIER-BYTE INSTRUCTIE
Bestaat uit twee bytes opcode en twee databytes of twee bytes opcode, één databyte en één byte opcode.

Voorbeelden:

LD IY,FF00	1 1 1 1 1 1 0 1	opcode
	0 0 1 0 0 0 0 1	opcode
	0 0 0 0 0 0 0 0	data (laag deel)
	1 1 1 1 1 1 1 1	data (hoog deel)
RLC (IX+0F)	1 1 0 1 1 1 0 1	opcode
	1 1 0 0 1 0 1 1	opcode
	0 0 0 0 1 1 1 1	data (verplaatsing)
	0 0 0 0 0 1 1 0	opcode

In de bit test-, set- en reset-instructies voor geïndexeerd geadresseerde operanden bevindt zich in het laatste opcode-byte een drie-bit veld om één van de acht bits van de operand aan te wijzen.
Voorbeeld:

SET b,(IY+0F)	1 1 1 1 1 1 0 1	opcode
	1 1 0 0 1 0 1 1	opcode
	0 0 0 0 1 1 1 1	data (verplaatsing)
	1 1 < b > 1 1 0	opcode (in b bitcode)

4. Adresseringswijzen

De data waarop een instructie betrekking heeft staat in één van de interne registers, in het geheugen of een I/O-apparaat. Adresseren is het aanwijzen van de plaats waar de data zich bevindt of waar de data naar toe moet.
In een instructie kunnen meerdere adresseringswijzen voorkomen: voor de bron of bronnen waar de data vandaan komt en voor de bestemming, de plaats waar de data heen moet.
De Z80 kent de volgende adresseringswijzen:

REGISTERADRESSERING
De data bevindt zich in één van de Z80-registers. Dat register wordt vaak aangewezen door een registerveld in de opcode.

Voorbeeld:

INC r	opcode:	0 0 < r > 1 0 0

De drie bits voor r bepalen op welk van de registers A, B, C, D, E, H, L de instructie wordt uitgevoerd.

Voorbeeld:

LD r,r'	opcode:	0 1 < r > < r'>

Zowel bestemming als bron, r en r', zijn register-geadresseerd. Twee drie-bits velden in de opcode bepalen om welk register het gaat.

ONMIDDELLIJKE ADRESSERING
De data staat in het geheugen direct achter de opcode. Als de opcode naar het instructieregister is gebracht, wijst de PC het adres van de data aan.

Voorbeeld:

LD r,0F	opcode:	0 0 < r > 1 1 0
	data:	0 0 0 0 1 1 1 1

In dit voorbeeld is de bron onmiddellijk en de bestemming register-geadresseerd.

ONMIDDELLIJKE UITGEBREIDE ADRESSERING
Het principe is gelijk aan dat van de onmiddellijke adressering met het verschil dat het hier gaat om 16-bit data voor het laden van de registerparen.

Voorbeeld:

LD ss,00FF	opcode:	0 0 s s 0 0 0 1
	data:	1 1 1 1 1 1 1 1
	data:	0 0 0 0 0 0 0 0

De bron is onmiddellijk uitgebreid geadresseerd. Het veld ss in de opcode wijst één van de registerparen BC, DE, HL of SP aan. De bestemming is dus register-geadresseerd.

REGISTER INDIRECTE ADRESSERING
In één van de registerparen staat het adres waarop zich de data bevindt, of waar de data naar toe moet.

Voorbeeld:

LD r,(HL)	opcode:	0 1 < r > 1 1 0

De bestemming r is register-geadresseerd.

UITGEBREIDE ADRESSERING
De instructie bevat een 16-bit adres, waarop data staat waar naar toe moet worden gesprongen of waarheen gesprongen wordt met een JP- of CALL-instructie.
Voorbeeld:

LD A,(00FF)	opcode:	0 0 1 1 1 0 1 0
	adres:	1 1 1 1 1 1 1 1
	adres:	0 0 0 0 0 0 0 0

De bestemming van A is register-geadresseerd. Uitgebreide adressering wordt vaak directe adressering genoemd.

GEINDEXEERDE ADRESSERING
Deze lijkt erg veel op de register indirecte adressering. Eén van de index-registerparen bevat een adres. Daarbij moet echter een verplaatsing worden opgeteld om het juiste adres van de data te vinden.

Voorbeeld:

AND (IY+0F)	opcode:	1 1 1 1 1 1 0 1
	opcode:	1 0 1 0 0 1 1 0
	verplaatsing:	0 0 0 0 1 1 1 1

De verplaatsing wordt opgevat als tweecomplement-getal. De bestemming, in dit geval de accumulator waarin het eindresultaat van de bewerking komt, is impliciet geadresseerd.

GEMODIFICEERDE PAGINA-NUL ADRESSERING
Pagina-nul adressering houdt in dat voor het aanwijzen van de geheugenlocatie waarin zich de data bevindt maar één byte wordt gebruikt en wel voor het lagere deel van het adres. Het hogere deel van het adres is altijd nul. Bij de Z80 is alleen een gemodificeerde pagina-nul adressering mogelijk met RST (re-start).

Voorbeeld:

RST p	opcode:	1 1 < p > 1 1 1

De drie voor p gereserveerde bits wijzen naar één van de acht adressen waar naar toe kan worden gesprongen.

RELATIEVE ADRESSERING
Alleen mogelijk bij relatieve sprongen. De instructie bevat een 8-bit getal dat bij de inhoud van de PC wordt opgeteld om het volledige 16-bit adres te krijgen waar naar toe moet worden gesprongen.

Voorbeeld:

JR 04	opcode:	0 0 0 1 1 0 0 0
	verplaatsing:	0 0 0 0 0 1 0 0

De verplaatsing wordt opgevat als tweecomplement-getal. De waarde van de PC is die na het ophalen van de verplaatsing. PC wijst naar de opcode van de volgende instructie.

IMPLICIETE ADRESSERING
Eén of meer registers worden automatisch beschouwd als bron of bestemming. In de instructie wordt niet naar een adres verwezen.

Voorbeeld:

EXX	opcode:	1 1 0 1 1 0 0 1

BIT-ADRESSERING
Wijst naar een enkel bit in een register- of geheugenlocatie.

Voorbeeld:

SET b,(HL)	opcode:	1 1 0 0 1 0 1 1
	opcode:	1 1 < b > 1 1 0

5. Het programmeren van de Z80

5.1 OVEREENKOMST MET DE INTEL 8080

De Z80 van Zilog is de 'opvolger' van de Intel 8080-microprocessor. De Z80 heeft weliswaar een veel grotere instructieset maar de object-codes van de 8080 zijn compatibel met die van de Z80, ook al is de source-code meestal anders. De instructie ADD HL,DE doet op de Z80 hetzelfde als de instructie DAD D op de Intel 8080. Beide hebben de object-code 19H.
Bovenstaande wil zeggen dat een Intel 8080-programma op de Z80 kan worden gedraaid. Omdat de Z80 veel meer instructies kent, is het omgekeerde niet mogelijk, tenzij in het Z80-programma alleen gebruik wordt gemaakt van instructies die ook de Intel 8080 kent. De toevoegingen op de 8080-set zijn in het algemeen traag en lang.
In afbeelding 5.1 en 5.2 (volgende pagina's) staan de equivalenten van de Z80- en Intel 8080-instructies.

5.2 BREEDTE DATABUS

De Z80 heeft een 8-bit databus. Uitwisseling met het geheugen van 16-bit data voor bijvoorbeeld stapelmanipulaties of het laden van de registerparen, gebeurt in twee fasen. Algemeen hierbij is dat eerst de laagste 8 bits (bit 0 t/m bit 7) worden verplaatst en daarna de 8 hoogste bits (bit 8 t/m bit 15). Uitzondering hierop is de PUSH-instructie.

5.3 INSTRUCTIES

In hoofdstuk 8 staat de volledige instructieset van de Z80 op alfabetische volgorde van mnemonic. In het nu volgende worden de instructies per groep behandeld.
Bij elke groep zijn de instructies in tabelvorm afgebeeld. De opcodes in de grijsgetinte vlakken komen overeen met die van de Intel 8080.

Z80	8080	Z80	8080	Z80	8080
ADC A,(HL)	ADC M	EX (SP),HL	XTHL	OR n	ORI [B2]
ADC A,n	ACI [B2]	HALT	HLT	OR r	ORA r
ADC A,r	ADC r	IN A,(n)	IN [B2]	OR (HL)	ORA M
ADD A,(HL)	ADD M	INC BC	INX B	OUT (n),A	OUT [B2]
ADD A,n	ADI [B2]	INC DE	INX D	POP AF	POP PSW
ADD A,r	ADD r	INC HL	INX H	POP BC	POP B
ADD HL,BC	DAD B	INC r	INR r	POP DE	POP D
ADD HL,DE	DAD D	INC SP	INX SP	POP HL	POP H
ADD HL,HL	DAD H	INC (HL)	INR M	PUSH AF	PUSH PSW
ADD HL,SP	DAD SP	JP C,nn	JC [B2][B3]	PUSH BC	PUSH B
AND n	ANI [B2]	JP M,nn	JM [B2][B3]	PUSH DE	PUSH D
AND r	ANA r	JP NC,nn	JNC [B2][B3]	PUSH HL	PUSH H
AND (HL)	ANA M	JP nn	JMP [B2][B3]	RET	RET
CALL C,nn	CC [B2][B3]	JP NZ,nn	JNZ [B2][B3]	RET C	RC
CALL M,nn	CM [B2][B3]	JP P,nn	JP [B2][B3]	RET M	RM
CALL NC,nn	CNC [B2][B3]	JP PE,nn	JPE [B2][B3]	RET NC	RNC
CALL nn	CALL	JP PO,nn	JPO [B2][B3]	RET NZ	RNZ
CALL NZ,nn	CNZ [B2][B3]	JP Z,nn	JZ [B2][B3]	RET P	RP
CALL P,nn	CP [B2][B3]	JP (HL)	PCHL	RET PE	RPE
CALL PE,nn	CPE [B2][B3]	LD A,(DE)	LDAX	RET PO	RPO
CALL PO,nn	CPO [B2][B3]	LD A,(nn)	LDA [B2][B3]	RET Z	RZ
CALL Z,nn	CZ [B2][B3]	LD DE,nn	LXID, [B2][B3]	RLA	RAL
CCF	CMC	LD SP,nn	LXI SP,[B2][B3]	RLCA	RLC
CP r	CMP r	LD (BC),A	STAX B	RRA	RAR
CP (HL)	CMP M	LD (DE),A	STAX D	RRCA	RRC
CPL	CMA	LD (HL),r	MOV M,r	RST P	RST P
CP n	CPI [B2]	LD (nn),A	STA [B2][B3]	SBC A,(HL)	SBB M
DAA	DAA	LD (nn),HL	SHLD [B2][B3]	SBC A,n	SBI [B2]
DEC BC	DCX B	LD A,(BC)	LDAX B	SBC A,r	SBB r
DEC DE	DCX D	LD BC,nn	LXIB [B2][B3]	SCF	STC
DEC HL	DCX H	LD HL,(nn)	LHLD [B2][B3]	SUB n	SUI [B2]
DEC r	DCR r	LD HL,nn	LXIH [B2][B3]	SUB r	SUB r
DEC SP	DCX SP	LD r,(HC)	MOV r,M	SUB (HL)	SUB M
DEC (HL)	DCR M	LD r,n	MVI r,[B2]	XOR n	XRI [B2]
DI	DI	LD r,r'	MOVE r1,r2	XOR r	XRA r
EI	EI	LD SP,HL	SPHL	XOR (HL)	XRA M
EX DE,HL	XCHG	NOP	NOP		

Afb. 5.1. Z80/8080-equivalenten.

5.3.1 De laadinstructies

De algemene vorm is: LD bestemming, bron. De instructie wijzigt de inhoud van de bron niet. In feite wordt de inhoud van de bron in de bestemming gekopieerd. Laadinstructies behelzen zowel 8- als 16-bit data. Het gebruik van haakjes in de source-code duidt op de inhoud van het adres dat tussen de haakjes staat.

Voorbeelden:

LD A,E	laadt de accumulator met de inhoud van register E.
LD A,(HL)	laadt de accumulator met de inhoud van het adres dat in HL staat.
LD (nn),A	laadt naar adres nn de inhoud van de accumulator.

8080	Z80	8080	Z80	8080	Z80
ACI [B2]	ADC A,n	IN [B2]	IN A,(n)	POP H	POP HL
ADC M	ADC A,(HL)	INR M	INC (HL)	POP PSW	POP AF
ADC r	ADC A,r	INR r	INC r	PUSH B	PUSH BC
ADD M	ADD A,(HL)	INX B	INC BC	PUSH D	PUSH DE
ADD r	ADD A,r	INX D	INC DE	PUSH H	PUSH HL
ADI [B2]	ADD A,n	INX H	INC HL	PUSH PSW	PUSH AF
ANA M	AND (HL)	INX SP	INC SP	RAL	RLA
ANA r	AND r	JC [B2][B3]	JP C,nn	RAR	RRA
ANI [B2]	AND n	JM [B2][B3]	JP M,nn	RC	RET C
CALL	CALL nn	JMP [B2][B3]	JP nn	RET	RET
CC [B2][B3]	CALL C,nn	JNC [B2][B3]	JP NC,nn	RLC	RLCA
CM [B2][B3]	CALL M,nn	JNZ [B2][B3]	JP NZ,nn	RM	RET M
CMA	CPL	JP [B2][B3]	JP P,nn	RNC	RET NC
CMC	CCF	JPE [B2][B3]	JP PE,nn	RNZ	RET NZ
CMP M	CP (HL)	JPO [B2][B3]	JP PO,nn	RP	RET P
CMP r	CP r	JZ [B2][B3]	JP Z,nn	RPE	RET PE
CNC [B2][B3]	CALL NC,nn	LDA [B2][B3]	LD A,(nn)	RPO	RET PO
CNZ [B2][B3]	CALL NZ,nn	LDAX B	LD A,(BC)	RRC	RRCA
CP [B2][B3]	CALL P,nn	LDAX D	LD A,(DE)	RST	RST P
CPE [B2][B3]	CALL PE,nn	LHLD [B2][B3]	LD HL,(nn)	RZ	RET Z
CPI [B2]	CP n	LXIB [B2][B3]	LD BC,nn	SBB M	SBC A,(HL)
CPO [B2][B3]	CALL PO,nn	LXID [B2][B3]	LD DE,nn	SBB r	SBC A,r
CZ [B2][B3]	CALL Z,nn	LXI H [B2][B3]	LD HL ,nn	SBI [B2]	SBC A,n
DAA	DAA	LXI SP [B2][B3]	LD SP,nn	SHLD [B2][B3]	LD (nn),HL
DAD B	ADD HL,BC	MOV M,r	LD (HL),r	SPHL	LD SP,HL
DAD D	ADD HL,DE	MOV r,M	LD r,(HL)	STA [B2][B3]	LD (nn),A
DAD H	ADD HL,HL	MOV r1,r2	LD r,r'	STAX B	LD (BC),A
DAD SP	ADD HL,SP	MVI M	LD (HL),n	STAX D	LD (DE),A
DCR M	DEC (HL)	MVI r [B2]	LD r,n	STC	SCF
DCR r	DEC r	NOP	NOP	SUB M	SUB (HL)
DCX B	DEC BC	ORA M	OR (HL)	SUB r	SUB r
DCX D	DEC DE	ORA r	OR r	SUI [B2]	SUB n
DCX H	DEC HL	ORI [B2]	OR n	XCHG	EX DE,HL
DCX SP	DEC SP	OUT [B2]	OUT (n),A	XRA M	XOR (HL)
DI	DI	PCHL	JP (HL)	XRA r	XOR r
EI	EI	POP B	POP BC	XRI [B2]	XOR n
HALT	HLT	POP D	POP DE	XTHL	EX (SP),HL

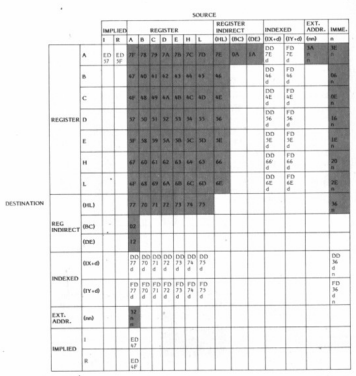
Afb. 5.2. 8080/Z80-equivalenten.

LD HL,nn laadt het L-register met het byte volgend op de opcode en, na verhoging van de PC, het H-register met het daarop volgende byte.

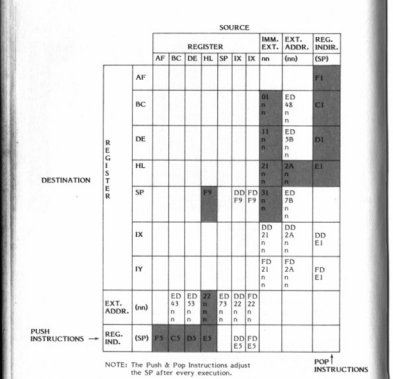
LD (nn),HL laadt het L-register naar adres nn en het H-register naar adres nn+1.

Van de 8-bit registers kan alleen de accumulator geladen of weggeschreven worden met een direct opgegeven adres of via een adres in de registerparen BC en DE. Bij de overige 8-bit registers moet het adres in HL of een index-register staan. HL is dus bijzonder geschikt als adrespointer.
De index-instructies zijn lang en traag. Bovendien moet de offset, de verplaatsing, van tevoren worden opgegeven. Voor het

zoeken in tabellen kan de basis van de tabel in DE en de tijdens het runnen van het programma nog te bepalen offset in HL. ADD HL,DE geeft dan in HL de juiste plaats in de tabel.



Afb. 5.3. 8-bits laadgroep.



Afb. 5.4. 16-bits laadgroep.

Enkele bijzondere 16-bit laadinstructies zijn de stapelmanipulaties PUSH en POP. De algemene vorm is:

PUSH registerpaar

of

POP registerpaar

PUSH DE bijvoorbeeld zet de inhoud van registerpaar DE op de stapel, ofwel laadt de inhoud van DE naar het adres waarnaar de inhoud van SP (stack pointer) wijst en wel op de volgende manier.
Eerst verlaagt de Z80 de inhoud van SP, die altijd naar de laatst bezette plaats van de stapel wijst, met één. De inhoud van D wordt gekopieerd in de geheugenlocatie waarnaar SP wijst. De Z80 verlaagt nogmaals de inhoud van SP en laadt de inhoud van E naar de geheugenlocatie waarnaar SP nu wijst.
POP DE doet het omgekeerde. De inhoud van de geheugenlocatie waarnaar SP wijst wordt gekopieerd in E. De Z80 verhoogt nu de inhoud van SP met één en kopieert de inhoud van de geheugenlocatie waarnaar SP nu wijst in D. SP wordt nogmaals verhoogd en wijst weer naar de laatst bezette plaats van de stapel.
De instructies PUSH DE, POP BC laden via de stapel het BC-registerpaar met de inhoud van DE. Hetzelfde zou zijn gebeurd met LD B,D en LD C,E.
Afbeeldingen 5.3 en 5.4 geven een overzicht van de 8- en 16-bit laadinstructies.

5.3.2 Register-verwisselingen

De instructies verwisselen inhouden van registerparen, die van een registerpaar met de top van de stapel of die van een registerpaar met de inhoud van het gelijknamige registerpaar uit de alternatieve registerset. Een bijzondere instructie is EXX, die de inhouden van BC, DE en HL verwisselt met die van BC', DE' en HL' uit de alternatieve registerset.
De algemene vorm:

EX registerpaar , registerpaar

of

EX (SP),registerpaar

Dankzij de instructie EX DE,HL is DE gemakkelijk te gebruiken als tweede adrespointer.

		IMPLIED ADDRESSING				
		AF'	BC', DE' & HL'	HL	IX	IY
IMPLIED	AF	08				
	BC, DE & HL		D9			
	DE			EB		
REG. INDIR.	(SP)			E3	DD E3	FD E3

Afb. 5.5. Exchanges.

5.3.3 Blokverplaatsing
Met blokverplaatsing-opdrachten kunnen een groot aantal opeenvolgende bytes met een enkele of een paar instructies in het geheugen worden verplaatst. Alle instructies gebruiken de registers BC, DE en HL.

HL wijst naar het adres van het te verplaatsen byte.
DE wijst naar het adres waar het byte heen moet.
BC wordt gebruikt als teller en wijst het aantal te verplaatsen bytes aan. Aangezien BC een 16-bit register is kan in één keer maximaal 64 Kbyte worden verplaatst (als de waarde van BC bij aanvang nul is).

Alle registers moeten zijn geladen voor de blokverplaatsing-instructie. Er zijn vier instructies mogelijk:

LDI (load increment) verplaatst het byte van (HL) naar (DE) waarna HL en DE met één worden verhoogd en teller BC met één wordt verlaagd. Is BC nu nul dan

wordt de P/V-vlag gereset.
LDD (load decrement) doet hetzelfde maar verlaagt de inhoud van HL en DE.

			SOURCE: REG. INDIR.	SOURCE: (HL)
			Reg HL points to source Reg DE points to destination Reg BC is byte counter	
DESTINATION	REG. INDIR.	(DE)	ED A0 'LDI' – Load (DE)⇐(HL) Inc HL & DE, Dec BC	
			ED B0 'LDIR' – Load (DE)⇐(HL) Inc HL & DE, Dec BC, Repeat until BC = 0	
			ED A8 'LDD' – Load (DE)⇐(HL) Dec HL & DE, Dec BC	
			ED B8 'LDDR' – Load (DE)⇐(HL) Dec HL & DE, Dec BC, Repeat until BC = 0	

Afb. 5.6. Blokverplaatsing.

LDIR (load increment repeat) herhaalt automatisch de LDI-instructie tot de inhoud van BC nul is.
LDDR (load decrement repeat) herhaalt automatisch de LDD-instructie tot de inhoud van BC nul is.

LDIR begint met de verplaatsing bij het laagste adres van het blok en LDDR met het hoogste. Of LDIR of LDDR binnen een programma worden gebruikt hangt sterk samen met de mogelijkheid dat het blok zichzelf gedeeltelijk overschrijft.

5.3.4 Blokvergelijking
Met blokvergelijking-opdrachten is het mogelijk om met één of een paar instructies een groot aantal opeenvolgende bytes in het geheugen te doorzoeken op de aanwezigheid van een bepaalde byte. Alle instructies gebruiken de accumulator en de registerparen HL en BC.

De inhoud van A is gelijk aan het te vinden byte.

HL bevat het adres van het eerste te onderzoeken byte.
BC bevat het aantal te onderzoeken bytes.

Deze registers moeten voor de blokvergelijking-instructie zijn geladen. Aangezien BC een 16-bit registerpaar is kunnen met één instructie 64 Kbytes worden doorzocht. De volgende instructies zijn mogelijk:

CPI (compare increment) trekt het byte in (HL) af van de accumulator en set de zeroflag als het resultaat nul is. De inhoud van de accumulator blijft onveranderd. Vervolgens wordt de inhoud van HL met één verhoogd en die van teller BC met één verlaagd. Is BC nu nul dan wordt de P/V-vlag gereset.

CPD (compare decrement) doet hetzelfde maar verlaagt de inhoud van HL.

CPIR (compare increment repeat) herhaalt de CPI-instructie net zolang tot het byte in (HL) gelijk is aan A of teller BC nul is.

CPDR (compare decrement repeat) doet hetzelfde met de CPD-instructie.

SEARCH LOCATION	HL points to location in memory to be compared with accumulator contents BC is byte counter
REG. INDIR. (HL)	
ED A1	'CPI' Inc HL, Dec BC
ED B1	'CPIR', Inc HL, Dec BC repeat until BC = 0 or find match
ED A9	'CPD' Dec HL & BC
ED B9	'CPDR' Dec HL & BC Repeat until BC = 0 or find match

Afb. 5.7. Blokvergelijking.

Een test van de zeroflag na de instructie moet uitmaken of het

gezochte byte al dan niet is gevonden. Is het gevonden dan wijst HL niet de juiste geheugenlocatie aan maar is, afhankelijk van de instructie, met één verhoogd of verlaagd. Bij CPIR bevindt het gezochte byte zich in de geheugenlocatie HL—1 en bij CPDR in HL+1.

5.3.5 8-Bit rekenkundige en logische instructies

Bij de rekenkundige instructies optellen en aftrekken, met of zonder carry, bevat de accumulator één van de twee operanden en na de instructie het resultaat. Behalve bij SUB s moeten beide operanden in de instructie worden gespecificeerd. Dus ADD A,(HL) maar SUB 14.

ADC A,B	inhoud accumulator	0010 0100
	inhoud B	0001 0100
	inhoud carry	1 +
	resultaat	0011 1001

Bij de vergelijk-instructie staat één van de te vergelijken bytes in de accumulator. Het andere byte moet in de instructie worden gespecificeerd. Het resultaat van de vergelijking beïnvloedt alleen de vlaggen. In feite komt het uitvoeren van de CP s (compare)-instructie er op neer dat operand s van de inhoud van de accumulator wordt afgetrokken. Het resultaat wordt genegeerd, de vlagbeïnvloeding niet.
Ook tot de rekenkundige instructies behoren increment en decrement. Een in de instructie te specificeren operand (INC m of DEC m) wordt met één verhoogd of verlaagd. Het resultaat komt op de plaats van de oorspronkelijke operand.
De logische instructies voeren een AND, een OR of een XOR uit tussen de inhoud van de accumulator en een in de instructie te specificeren operand. Het resultaat komt in de accumulator.
Rekenkundige en logische instructies kunnen ook voor andere doeleinden worden gebruikt. SUB A of XOR A laden de accumulator sneller met 0 dan LD A,0 maar beïnvloeden tevens de vlaggen. ADD A,A geeft een snelle verschuiving naar links.
Er zijn nog een vijftal algemeen rekenkundige instructies die werken met de accumulator of de carry-vlag.
CCF (complement carry flag) en SCF (set carry flag) complementeren of setten de carry-vlag. Het resetten van de carry kan worden gedaan door een logische AND van de accumulator met

SOURCE

	REGISTER ADDRESSING							REG. IND.	INDEXED		IMMED.
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
'ADD'	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
ADD w CARRY 'ADC'	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUBTRACT 'SUB'	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SUB w CARRY 'SBC'	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
'AND'	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
'XOR'	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
'OR'	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
COMPARE 'CP'	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INCREMENT 'INC'	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DECREMENT 'DEC'	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

Afb. 5.8. 8-Bits rekenkundig en logisch.

zichzelf (AND A). De inhoud van A wordt hierdoor niet beïnvloed. De overige drie zijn:

- CPL (complement) complementeert de inhoud van de accumulator.

- NEG (negate) geeft de twee-complements-voorstelling van de negatief gemaakte inhoud van de accumulator.

accumulator voor NEG: 0101 0110 56H = 86 decimaal
accumulator na NEG: 1010 1010 AAH = 170 decimaal of −86 in twee-complement

Wat betreft het twee-complement het volgende. De programmeur bepaalt wat de inhoud van een byte op een gegeven moment voorstelt. Het kan een cijfer zijn, een letter of een teken uit welk coderingssysteem dan ook. Stelt het een getal voor waarop rekenkundige operaties worden uitgevoerd, dan gelden daarvoor de binaire rekenregels. Aangezien een byte zelf geen plus- of minteken heeft moet, als er met positieve en negatieve getallen wordt gewerkt, het teken in de byte worden verwerkt.
Voor het weergeven van negatieve getallen in binaire vorm wordt de twee-complement-methode gebruikt. Een negatief getal als -3 krijgt voor een 8-bits formaat de waarde 2^8-3=256-3=253 en 16 de waarde 256-16=240. In een 16-bits formaat zou dit 2^{16}-3 respectievelijk 2^{16}-16 zijn.
Een deel van de in één byte weer te geven getallen wordt als negatief beschouwd en wel die waarvan het hoogste bit een 1 is. Dit hoogste bit heet tekenbit en is 0 voor positieve getallen. Deze indeling levert positieve getallen op tot en met 127 (0111 1111) en negatieve tot en met -128 (1000 0000=256-128=128). Een getal, opgeteld bij de inverse ervan levert 0 op. In feite 256, maar het negende bit dat dit aangeeft kan niet in het 8-bits getal maar komt in de carry.

twee-complement van -3 = 253 1111 1101
+3 0000 0011 +
10000 0000

En -3+16 wordt berekend als 256-3+16=256+13, wat in 8-bits formaat 13 oplevert.
Een andere manier om het twee-complement van een getal te berekenen is alle bits inverteren, wat neerkomt op het getal aftrekken van 255, en het resultaat met 1 verhogen, wat hetzelfde is als het getal van 256 aftrekken.
Als bij een rekenkundige bewerking, bijvoorbeeld het optellen van twee grote positieve getallen waardoor bit 7 één wordt en

het resultaat dus negatief zou zijn, het teken ten onrechte verandert, wordt de overflow-vlag geset. In feite wordt de overflow-vlag geset als er in een berekening een carry is van bit 6 naar bit 7 maar niet naar de carry-vlag of geen carry van bit 6 naar bit 7 maar wel naar de carry-vlag.
Tot slot nog een tweetal voorbeelden:

```
twee-complement  —127  1000 0001   binair 129
                 +126  0111 1110  + binair 126
twee-complement  —1    1111 1111   binair 255

                 +127  0111 1111   binair 127
                 +127  0111 1111  + binair 127
twee-complement  —2    1111 1110   binair 254
```

In het tweede voorbeeld is er een carry van bit 6 naar bit 7 maar niet naar het carry-bit. Het tekenbit verandert dus het resultaat is fout en de overflow-vlag wordt geset. Merk op dat het resultaat wel correct is als de getallen worden beschouwd als gewone binaire getallen. Of getallen als twee-complement worden beschouwd of niet is, zoals gezegd, een zaak van de programmeur.

- DAA (decimal adjust accumulator) corrigeert na een rekenkundige bewerking de inhoud van de accumulator voor BCD (binary coded decimal).

De Z80 bepaalt de correctie aan de hand van carry, half carry en subtract-vlag. De correctie is juist na de instructies ADD, ADC, SUB, SBC en NEG. INC en DEC beïnvloeden de carry-vlag niet. Correcties na deze instructies kunnen fouten opleveren. Alternatief voor INC is ADD A,1 en vervolgens DAA.
Net zoals het twee-complement een opvatting is van hetgeen een byte voorstelt, is dat het geval bij BCD. Daarbij vormen linker en rechter vier bits, ofwel linker- en rechter-nibble, een decimaal getal van twee tekens.

```
BCD-voorstelling 39:    0011 1001
                  (3)   (9)
```


Aangezien decimale getallen een waarde van 0 t/m 9 hebben en 4 bits een getal van 0 t/m 15 kunnen bevatten, moet na een rekenkundige bewerking meestal een correctie volgen. De correctiefactor voor de linker en/of rechter vier bits is 15—9=6.

```
BCD-voorstelling 39     0011 1001   binair 57
BCD-voorstelling 14     0001 0100 + binair 20
BCD-voorstelling 47     0100 1101   binair 77
```

De binaire optelling klopt, maar het getal in de rechter 4 bits is te groot voor een enkel decimaal getal, namelijk 13.

```
BCD-voorstelling 47     0100 1101
correctie 06            0000 0110 +
BCD-voorstelling 53     0101 0011
```

Zodra het resultaat van de optelling van de twee rechter byte-helften groter is dan 9 of als er een carry van de rechter naar de linker byte-helft is gegaan (half carry-vlag) moet bij de rechter byte-helft van het resultaat 6 worden opgeteld.

Decimal Adjust Acc, 'DAA'	27
Complement Acc, 'CPL'	2F
Negate Acc, 'NEG' (2's complement)	ED 44
Complement Carry Flag, 'CCF'	3F
Set Carry Flag, 'SCF'	37

Afb. 5.9. Algemene AF-operaties.

Is er een carry van de linker byte-helft naar de carry-bit gegaan, of is het resultaat van de optelling van beide linker byte-helften groter dan 9, dan moet bij de linker byte-helft 6 worden opgeteld.
Bij aftrekken is de correctie anders. Aangezien van twee BCD-getallen wordt uitgegaan kunnen de byte-helften niet op dezelfde

manier als bij de optelling te groot worden. Onderscheid welke bewerking aan de DAA is voorafgégaan, wordt gemaakt met behulp van de subtract-vlag.
Bij een borrow van de linker naar de rechter byte-helft is er teveel geleend, namelijk 16 in plaats van 10. Van het resultaat moet ter correctie 6 worden afgetrokken. Hetzelfde doet zich voor bij een borrow van het carry-bit naar de linker byte-helft.

5.3.6 16-Bit rekenkundig
Deze instructies werken met de registerparen en kunnen onder andere worden gebruikt voor het berekenen van adressen in tabellen. Voor optellen en aftrekken fungeert HL als 16-bits accumulator, dat wil zeggen HL bevat voor de berekening één van de operanden en erna het resultaat. Aftrekken zonder carry is niet mogelijk.

DESTINATION		SOURCE BC	DE	HL	SP	IX	IY
'ADD'	HL	09	19	29	39		
	IX	DD 09	DD 19		DD 39	DD 29	
	IY	FD 09	FD 19		FD 39		FD 29
ADD WITH CARRY AND SET FLAGS 'ADC'	HL	ED 4A	ED 5A	ED 6A	ED 7A		
SUB WITH CARRY AND SET FLAGS 'SBC'	HL	ED 42	ED 52	ED 62	ED 72		
INCREMENT 'INC'		03	13	23	33	DD 23	FD 23
DECREMENT 'DEC'		08	18	28	38	DD 2B	FD 2B

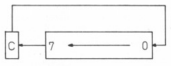
Afb. 5.10. 16-Bits rekenkundig.

Voor optellingen zonder carry kunnen de index-registers als accumulator fungeren.

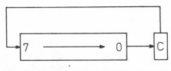
8 -bit rekenkundig:	ADD A,B	INC C
16-bit rekenkundig:	ADD HL,BC	INC DE

5.3.7 Rotaties en verschuivingen
Rotaties of verschuivingen kunnen worden uitgevoerd met elk van de 8-bit registers en, register indirect of geïndexeerd geadresseerd, met elke geheugenlocatie. De verschoven of geroteerde operand wordt teruggezet in de oorspronkelijke locatie.

- Rotaties
Rotaties kunnen door of langs het carry-bit plaatsvinden. In het eerste geval zijn de instructies voor het links en rechts roteren RL operand (rotate left) en RR operand (rotate right). Twee extra instructies geven een snelle rotatie voor de accumulator: RLA (rotate left accumulator) en RRA (rotate right accumulator). De vlagbeïnvloeding van deze twee instructies is echter anders. Bij al deze rotaties fungeert de carry als negende bit. Alle bits in de operand schuiven een plaats naar links of naar rechts. Het eruit vallende bit komt in de carry en het vrijkomende bit wordt opgevuld door de inhoud van de carry. Zie afbeeldingen 5.11 en 5.12.



Afb. 5.11. Roteer links (RL en RLA).

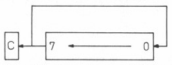


Afb. 5.12. Roteer rechts (RR en RRA).

Voor de rotaties langs de carry zijn de instructies RLC operand (rotate left circular) en RRC operand (rotate right circular).

Ook hier zijn extra instructies voor een snelle rotatie van de accumulator: RLCA (rotate left circular accumulator) en RRCA (rotate right circular accumulator). De vlagbeïnvloeding van deze twee instructies is echter anders. Bij al deze rotaties fungeert de carry niet als negende bit. Het door de operatie links of rechts uit de operand vallende bit neemt de vrijgekomen plaats aan de andere kant in en wordt gekopieerd in de carry. Zie afbeeldingen 5.13 en 5.14.

Afb. 5.13. Roteer naar rechts langs de carry (RRC en RRCA).



Afb. 5.14. Roteer links langs de carry (RLC en RLCA).

Dat er voor al deze rotaties, toegepast op de accumulator, twee verschillende instructies met twee verschillende opcodes zijn, zoals RLA en RL A, vindt zijn oorzaak in de instructieset van de 8080. Alle snelle instructies zoals RLA kwamen daarin voor. Voor de Z80 kwamen daar de andere bij, zoals RR m, waarbij m elk register kan zijn, dus ook A, en eveneens de geheugenlocaties aangewezen door (HL), (IX+d) en (IY+d).

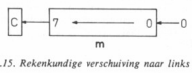
Voorbeelden:

```
RLC D          RR (HL)
```

- Verschuivingen

Verschuivingen zijn mogelijk met elk 8-bit register en, register indirect of geïndexeerd geadresseerd, met iedere geheugenplaats. De volgende drie verschuivingen laten zich onderscheiden:

- shift left arithmetic (rekenkundige verschuiving naar links)



Afb. 5.15. Rekenkundige verschuiving naar links (SLA).

Alle bits in de operand worden één plaats naar links geschoven. Het eruit vallende bit komt in de carry en het vrijkomende bit wordt nul. De bewerking komt in feite neer op een vermenigvuldiging met twee.
Voorbeeld:

```
inhoud accumulator:          0111 0001  113
na SLA A:                    1110 0010  226
```

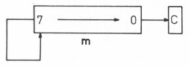
ADD A,A heeft hetzelfde resultaat als SLA A maar is korter, één byte opcode in plaats van twee, en sneller.
Nogmaals verschuiven naar links zou een foutief resultaat opleveren omdat daardoor bit 7 (1) in de carry zou komen. Gebruik maken van een registerpaar en de RL-instructie voorkomt dat.

Voorbeeld:

```
LD D,0          : D-register nul maken
LD E,226        : operand in E-register
SLA E           : bit 7 naar carry
RL D            : carry naar D-register

inhoud DE-registerpaar  :  0000 0000 1110 0010   226
na SLA E                :  0000 0000 1100 0100   196
in carry                :  1
na RL D                 :  0000 0001 1100 0100   452
```

- shift right arithmetic (rekenkundige verschuiving naar rechts)



Afb. 5.16. Rekenkundige verschuiving naar rechts (SRA).

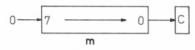
Alle bits in de operand worden één plaats naar rechts verschoven. Het eruit vallende bit komt in de carry. Bit 7 blijft gelijk. Het geheel komt neer op een deling door twee voor twee-complements getallen. Doordat bit 7 gelijk blijft, wordt het teken behouden.

Voorbeeld:

inhoud accumulator	:	1001 0000	-112
na SRA A	:	1100 1000	-56

Wat er gebeurt is het volgende. De twee-complements notatie voor het negatieve getal -X=256-X. Die voor -½X=256-½X. Een gewone verschuiving naar rechts zou opleveren ½(256-X)=128-½X. Door nu het tekenbit te kopiëren wordt bij de gewone verschuiving 128 opgeteld waardoor het resultaat is: 128+128-½X=256-½X. Zie voor meer over de twee-complements notatie de behandeling van de NEG-instructie in dit hoofdstuk.

- shift right logical (rekenkundige verschuiving naar rechts)



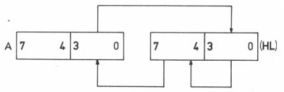
m

Afb. 5.17. Logische verschuiving naar rechts (SRL).

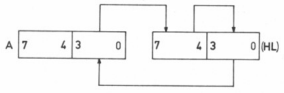
Alle bits in de operand worden één plaats naar rechts geschoven. Het eruit vallende bit komt in de carry en het vrijkomende bit wordt nul. De operatie komt neer op een deling door twee voor gewone binaire getallen.

- rotate digit (roteer byte-helften of nibbles)

De te gebruiken instructies zijn RLD (rotate left digit) en RRD (rotate right digit). Byte-helften worden geroteerd tussen de accumulator en de door HL aangewezen geheugenlocatie. Zie afbeeldingen 5.18 en 5.19.



Afb. 5.18. BCD-rotatie naar links (RLD).



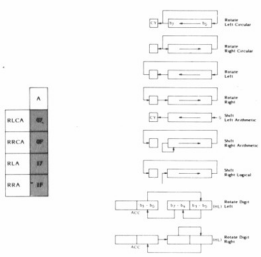
Afb. 5.19. BCD-rotatie naar rechts (RRD).

RRD bewerkstelligt het volgende. De lage byte-helft van de accumulator gaat naar de hoge byte-helft van de geheugenlocatie waarnaar HL wijst. De hoge byte-helft daarvan gaat naar de lage byte-helft en de lage byte-helft van HL gaat naar de lage byte-helft van de accumulator. Al deze verplaatsingen vinden in één keer plaats. De instructies worden toegepast bij het rekenen met BCD-getallen. Zie voor meer daarover de behandeling van de DAA-instructie in dit hoofdstuk.
Een toepassingsvoorbeeld is het omzetten van BCD-getallen naar ASCII-code. Bij de BCD-voorstelling staan in de byte-helften de decimale getallen 0 t/m 9. De ASCII-code daarvoor is 30 t/m 39 hexadecimaal.
In de geheugenlocatie waarnaar HL wijst staat de BCD-notatie voor het getal 58.

inhoud (HL)	0101 1000	58 in BCD
accumulator	0011 0000	30 hexadecimaal

TYPE OF ROTATE OR SHIFT	Source and Destination									
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
'RLC'	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
'RRC'	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
'RL'	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
'RR'	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
'SLA'	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
'SRA'	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
'SRL'	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
'RLD'								ED 6F		
'RRD'								ED 67		

Afb. 5.20. Rotaties en verschuivingen (eerste deel).



Afb 5.20. Rotaties en verschuivingen (tweede deel).

Na de RRD-instructie is de situatie:

inhoud (HL)	0000 0101	
accumulator	0011 1000	8 in ASCII-code, ofwel 38 hexadecimaal

En na de volgende RRD-instructie:

inhoud (HL)	1000 0000	
accumulator	0011 0101	5 in ASCII-code, ofwel 35 hexadecimaal

Een derde RRD-instructie brengt alles terug in de oorspronkelijke situatie:

inhoud (HL)	0101 1000
accumulator	0011 0000

De gevonden ASCII-codes moeten tussentijds worden weggezet. Op

dezelfde manier kunnen ASCII-codes worden omgezet in BCD-getallen.

5.3.8 Bitmanipulatie

De afzonderlijke bits van elk 8-bit register en, register indirect of geïndexeerd geadresseerd, van iedere geheugenplaats kunnen getest, geset of gereset worden. De instructies daarvoor zijn BIT (test), SET en RES (reset). In de instructie moeten bitnummer en de operand worden aangegeven. SET b,m bijvoorbeeld set bit b van operand m. Om bit 3 van de accumulator te resetten is de instructie SET 3,A nodig. De nummers van de bits zijn als volgt verdeeld:

inhoud byte : 0 0 1 1 0 1 0 1
bitnummer : 7 6 5 4 3 2 1 0

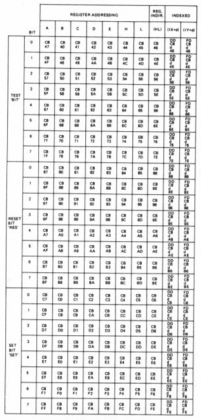
Het resultaat van de bit-test-instructie komt in de zero flag. Is het geteste bit nul dan wordt de zero flag geset, anders gereset. Na de testinstructie is de zero flag de inverse van het geteste bit.

5.3.9 Sprong, call en return

Sprongen en call's doorbreken het sequentiële verloop van een programma door de inhoud van de PC (program counter) te veranderen. Het verschil tussen een sprong en een call is dat de laatste voor de wijziging de inhoud van de PC op de stapel zet. De return-instructie, waarmee de door de call aangeroepen subroutine wordt afgesloten, haalt de oorspronkelijke waarde van de stapel en zet die terug in de PC. Het programma gaat verder vanaf de plaats waar het werd onderbroken.

- Sprongen

Er zijn relatieve en absolute sprongen. Beide kunnen voorwaardelijk en onvoorwaardelijk zijn. De instructie voor de absolute sprong is JP adres (jump). Het nieuwe 16-bit adres voor de PC moet in de instructie worden meegegeven. De voorwaardelijke absolute sprong heeft de vorm JP cc,adres waarbij cc een voorwaarde is die samenhangt met een vlagtest. Bijvoorbeeld JP Z,FEFE laadt adres FEFE in de PC als de zero flag geset is. De mogelijke voorwaarden zijn: C (carry), NC (non carry), Z (zero),



Afb. 5.21. Bitmanipulatie.

NZ (non zero), PO (pariteit oneven of geen overflow), PE (pariteit even of overflow), P (plus), M (min).
De onvoorwaardelijke absolute sprong kan ook register indirect zijn geadresseerd en heeft dan de vorm JP (HL), JP (IX), JP (IY). Het gespecificeerde register bevat het 16-bit adres dat in de PC wordt geladen.
Bij de relatieve sprong hoeft niet een volledig 16-bit adres te worden meegegeven maar een 8-bit verplaatsing. Deze verplaatsing is een twee-complement getal dat bij de momentele waarde van de PC wordt opgeteld. De instructie voor een relatieve sprong is dus één byte korter. Een relatieve sprong uitvoeren kost meer tijd dan een absolute.
De vorm is JR e (jump relative). Aangezien een 8-bit twee-complement getal een waarde kan hebben van +127 t/m -128 is dat het grootste bereik van de sprong. De uitwerking van JR -7 is dat van de inhoud van de PC, 7 wordt afgetrokken. Dat gebeurt op een moment dat de PC al naar de volgende instructie wijst en dus 2 hoger is dan aan het begin van JR e. De verplaatsing ten opzichte van deze laatste is dus +129 t/m -126.
De voorwaardelijke relatieve sprong heeft de vorm JR cc,e. De voorwaarde cc is beperkter dan bij de absolute sprong.
Mogelijkheden zijn: C (carry), NC (non carry), Z (zero), NZ (non zero). Een bijzondere relatieve sprong is DJNZ e. Deze verlaagt de inhoud van het als teller te gebruiken B-register en maakt dan een in e op te geven relatieve sprong als B hierdoor niet nul is geworden. Deze snelle korte instructie wordt veel gebruikt voor programmalussen.

- Call's
Call's onderscheiden zich van sprongen doordat de instructie eerst de inhoud van de PC op de stapel zet alvorens die met het nieuwe adres te laden. Op het moment waarop de PC op de stapel wordt gezet wijst deze naar de instructie volgend op de call.
Call's kunnen voorwaardelijk en onvoorwaardelijk zijn: CALL cc adres en CALL adres. De voorwaarden zijn dezelfde als die voor de absolute sprong.
CALL NZ,FEFE roept een subroutine die begint op adres FEFE aan als de zero flag niet geset is.

- Return
De return-instructie is de afsluiting van een subroutine en zet de top van de stapel in de PC zodat het programma verder gaat vanaf het punt waar het door de call werd onderbroken.

CONDITION

			UN-COND.	CARRY	NON CARRY	ZERO	NON ZERO	PARITY EVEN	PARITY ODD	SIGN NEG.	SIGN POS.	REG. B/0
JUMP "JP"	EXT.	nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JUMP "JR"	RELATIVE	PC+e	18 e-2	38 e-2	30 e-2	28 e-2	20 e-2					
JUMP "JP"		(HL)	E9									
JUMP "JP"	REG. INDIR.	(IX)	DD E9									
JUMP "JP"		(IY)	FD E9									
"CALL"	EXT.	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
DECREMENT B, JUMP IF NON ZERO DJNZ	RELATIVE	PC+e										10 e-2
RETURN "RET"	REGISTER INDIR.	(SP) (SP+1)	C9	D8		C8	C0	E8	E0	F8	F0	
RETURN FROM INT "RETI"	REG. INDIR.	(SP) (SP+1)	ED 4D									
RETURN FROM NON MASKABLE INT "RETN"	REG. INDIR.	(SP) (SP+1)	ED 45									

NOTE: CERTAIN FLAGS HAVE MORE THAN ONE FUNCTION; REFER TO SECTION 6.0 FOR DETAILS

Afb. 5.22. Jump-, call- en return-groep.

Er zijn drie return-instructies: RET, RETI (return from interrupt), RETN (return from non maskable interrupt). De beide laatste sluiten subroutines af die door een interrupt werden aangeroepen. De uitwerking is precies hetzelfde als van RET, de opcodes echter worden door de randapparatuur-chips herkend als het einde van een interrupt-service-routine.
RET kan als enige van de drie ook een voorwaardelijke instructie zijn. De vorm daarvan is RET cc. De voorwaarden zijn dezelfde als die voor een absolute sprong.

5.3.10 Restart
De instructies voeren een call uit naar één van de acht mogelijke pagina-nul adressen. Voordelen zijn de snelheid en geringe lengte van de instructie: met slechts één byte opcode wordt een volledige call uitgevoerd. Drie bits binnen de opcode selecteren één van de acht adressen. Nadeel is het geringe aantal beschikbare adressen.

		OP CODE	
CALL ADDRESS	0000_H	C7	'RST 0'
	0008_H	CF	'RST 8'
	0010_H	D7	'RST 16'
	0018_H	DF	'RST 24'
	0020_H	E7	'RST 32'
	0028_H	EF	'RST 40'
	0030_H	F7	'RST 48'
	0038_H	FF	'RST 56'

Afb. 5.23. Restart-groep.

5.3.11 Input en output
De Z80 heeft een groot aantal input- en output-instructies. Input of output van een enkel byte kan direct of register indirect geadresseerd geschieden. Vanzelfsprekend heeft gebruik

ervan geen zin als I/O memory mapped is georganiseerd.
Direct geadresseerd heeft de instructie voor input respectievelijk output de vorm IN A,(n) en OUT (n),A. Hierin is n een één-byte poortadres dat in de instructie moet worden meegegeven. Poortadres n wordt geplaatst op de lagere helft van de adresbus (A0 t/m A7) en de inhoud van de accumulator op de hogere helft. Bij input komt het van de poort ingelezen byte in de accumulator. Bij output wordt de inhoud van de accumulator naar de poort geschreven.
Register indirect geadresseerd heeft de instructie voor input respectievelijk output de vorm IN r,(C) en OUT (C),r. Hierin kan r elke van de 8-bit registers zijn. Het poortadres bevindt zich in register C en wordt op de lagere helft van de adresbus geplaatst. Op de hogere helft (A0 t/m A7) komt de inhoud van register B. Bij input komt het van de poort ingelezen byte in register r. Bij output wordt de inhoud van register r naar de poort geschreven. Om een byte van de door register C aangegeven poort in register H te lezen moet de instructie zijn: IN H,(C).
Een ander verschil tussen de twee soorten input- en output-instructies is de vlagbeïnvloeding. IN A,(n) en OUT (n),A beïnvloeden de vlaggen niet. De instructie IN r,(C) en OUT r,(C) beïnvloeden op de carry na alle vlaggen.

Behalve de input- en output-instructies voor een enkel byte zijn er ook blok-input en blok-output. Deze zijn gelijkvormig met de blok-laad- en blok-vergelijk-instructies.
De mogelijkheden zijn: INI (input increment), IND (input decrement), INIR (input increment repeat), INDR (input decrement repeat), OUTI (output increment), OUTD (output decrement), OTIR (output increment repeat), OTDR (output decrement repeat).
Bij al deze instructies staat het poortadres in register C. Register B fungeert als teller; er kunnen dus maximaal 256 bytes worden verplaatst (als B nul is aan het begin van de instructie).
Registerpaar HL wijst naar de geheugenlocatie waar het weg te schrijven byte bij een input-instructie naar toe moet of waar het bij een output-instructie vandaan komt. De registers moeten voor het gebruik van de instructies geladen zijn. INI leest een byte van de door register C aangewezen poort en plaatst het in de door registerpaar HL aangewezen geheugenlocatie. Daarna wordt de inhoud van register B, de teller, met één verlaagd en die van registerpaar HL met één verhoogd. IND doet hetzelfde,

behalve dan dat HL wordt verlaagd in plaats van verhoogd. OUTI en OUTD schrijven op dezelfde manier bytes uit de door registerpaar HL aangewezen geheugenlocatie naar de door register C aangewezen poort. Net als bij IN r,(C) en OUT (C),r wordt de inhoud van register C op de lagere helft van de adresbus geplaatst en die van register B op de hogere helft. De zero flag wordt geset als de inhoud van register B nul is aan het einde van de instructie. De repeat-instructies INIR, INDR, OTIR en

INPUT DESTINATION				SOURCE PORT ADDRESS: IMMED. (n)	SOURCE PORT ADDRESS: REG. INDIR. (c)
INPUT 'IN'	REG ADDRESS	A	DB n	ED 78
		B		ED 40
		C		ED 48
		D		ED 50
		E		ED 58
		H		ED 60
		L		ED 68
'INI'-INPUT & Inc HL, Dec B	REG. INDIR (HL)			ED A2
'INIR'-INPUT, Inc HL, Dec B, REPEAT IF B/0				ED B2
'IND'-INPUT & Dec HL, Dec B				ED AA
'INDR'-INPUT, DEC HL, Dec B, REPEAT IF B/0				ED BA

BLOCK INPUT COMMANDS

Afb. 5.24. Input.

OTDR doet hetzelfde als INI, IND, OUTI en OUTD en herhalen bovendien de instructies tot de inhoud van register B, de teller, nul is. Hiermee kunnen, in een instructie, 256 bytes van of naar een poort worden ingelezen of weggeschreven.

SOURCE		REGISTER A	B	C	D	E	H	L	REG. INDIR. (HL)	BLOCK OUTPUT COMMAND:
PORT DESTINATION ADDRESS	IMMED. (n)	D3 n	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69		ED A3
	REG. INDIR. (C)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69		ED B3
'OUT'	REG. INDIR. (C)									ED AB
'OUTI'-OUTPUT Inc HL, Dec b	REG. INDIR. (C)									ED A3
'OTIR'-OUTPUT, Inc HL, Dec B, REPEAT IF B/0	REG. INDIR. (C)									ED B3
'OUTD'-OUTPUT Dec HL & B	REG. INDIR. (C)									ED AB
'OTDR'-OUTPUT, Dec HL & B, REPEAT IF B/0	REG. INDIR. (C)									ED BB

Afb. 5.25. Output-groep.

5.3.12 Controle-instructies

De controle-instructies veranderen de werkwijze van de Z80. Afbeelding 5.26 geeft een overzicht van de mogelijkheden.

NOP (no operation) laat de Z80 gedurende 1 machinecyclus niets doen. HALT laat de Z80 NOP-instructies uitvoeren tot een interrupt-aanvraag wordt ontvangen. Beide instructies worden uitgebreid behandeld in de instructieset. Zie daarvoor hoofdstuk 8. De andere controle-instructies hebben betrekking op de interrupt-afhandeling. EI (enable interrupts) en DI (disable interrupts) staan respectievelijk maskeerbare interrupts toe en verbieden ze.

IM 0, IM 1 en IM 2 zetten de Z80 in de interrupt-mode 0, 1 of 2. Deze worden behandeld in de instructieset. Bovendien is aan de interrupts een apart hoofdstuk gewijd: hoofdstuk 7.

'NOP'	00	
'HALT'	76	
DISABLE INT '(DI)'	F3	
ENABLE INT '(EI)'	FB	
SET INT MODE 0 "IM0"	ED 46	8080A MODE
SET INT MODE 1 "IM1"	ED 56	CALL TO LOCATION 0038_H
SET INT MODE 2 "IM2"	ED 5E	INDIRECT CALL USING REGISTER I AND 8 BITS FROM INTERRUPTING DEVICE AS A POINTER.

Afb. 5.26. Controle-groep.

6. Vlaggen

De Z80 heeft voor de vlaggen een 8-bit register in de gewone zowel als in de alternatieve registerset. Van de 8 bits worden er maar 6 gebruikt. Daarvan zijn er 4 toegankelijk voor het programmeren. Het zijn:

6.1 CARRY FLAG

Rekenkundige instructies optellen en aftrekken beïnvloeden het carry-bit. Het bit wordt geset bij een carry of borrow in de berekening, anders gereset. Dit geldt eveneens voor CP s, NEG en DAA. Bij CP s wordt een 8-bit getal van de accumulatorinhoud afgetrokken. NEG trekt de inhoud van de accumulator af van nul. Bij DAA geeft het carry-bit een borrow of carry aan voor een packed-BCD-berekening.

Rotatie- en schuifinstructies gebruiken de carry als negende bit. XOR, AND en OR resetten de carry.

Er zijn twee speciale instructies voor het carry-bit. SCF (Set Carry Flag), set het carry-bit. CCF (Complement Carry Flag), complementeert de inhoud van het carry-bit.

6.2 ZERO FLAG

Bij een groot aantal instructies geeft het zero-bit aan of het resultaat al dan niet nul is. Is het resultaat nul dan wordt het zero-bit geset, anders gereset. Dat is eveneens het geval als bij IN r,(C) het ingelezen byte nul is.

Bit-test-instructies setten de zero flag als het aangewezen bit nul is, zoniet dan wordt de zero flag gereset. Ofwel: het zero-bit is het complement van het geteste bit.

Blok-input- en output-instructies setten het zero-bit als teller B nul is. Alle vergelijkingsinstructies CP s, CPI, CPIR, CPD, CPDR setten het zero-bit als het onderzochte byte gelijk is aan

de inhoud van de accumulator. Anders wordt het zero-bit gereset.

6.3 PARITY/OVERFLOW FLAG

Deze vlag geeft de pariteit van het resultaat bij logische operaties, verschuivingen en rotaties (behalve RLA, RLCA, RRA, RRCA), en bij DAA en IN r,(C). Bij even pariteit wordt de vlag geset, bij oneven pariteit gereset.
Na rekenkundige bewerkingen (behalve 16-bit ADD) fungeert de vlag als overflow-indicator. Is de vlag geset dan is het resultaat van de berekening te groot voor een twee-complement getal; dat wil zeggen voor een 8-bit register groter dan +127 of kleiner dan -128.
Bij de instructies LD A,I en LD A,R wordt de inhoud van de interrupt-flipflop IFF2 in de vlag gekopieerd.
Blokverplaats- en blokvergelijk-instructies gebruiken de vlag om aan te geven of de teller BC al dan niet nul is. De vlag is geset als de inhoud van BC ongelijk is aan nul, anders gereset. Aan het einde van LDIR en LDDR is de vlag dus altijd gereset.

6.4 SIGN FLAG

Het tekenbit is een kopie van bit 7 (most significant bit) van het resultaat van een bewerking. Bij twee-complements notatie is de vlag geset bij een negatief en gereset bij een positief resultaat.

De inhoud van de hierboven genoemde vier vlaggen kan worden gebruikt als voorwaarde bij sprongen, calls en returns. De voorwaarden zijn:

C	: carry (carry-bit geset)	JP, JR, CALL, RET
NC	: non carry (bit gereset)	JP, JR, CALL, RET
Z	: zero (bit geset)	JP, JR, CALL, RET
NZ	: non zero (bit gereset)	JP, JR, CALL, RET
PE	: pariteit even (bit geset)	JP, CALL, RET
PO	: pariteit oneven (bit gereset)	JP, CALL, RET
M	: min (bit geset)	JP, CALL, RET
P	: plus (bit gereset)	JP, CALL, RET

De twee volgende vlaggen kunnen niet als voorwaarde binnen een instructie worden gebruikt. Ze worden door de Z80 gebruikt bij

de DAA-instructie (Decimal Adjust Accumulator) voor het corrigeren van packed-BCD-berekeningen.

6.5 HALF CARRY FLAG

De vlag is geset als er een carry of borrow is van de vier hoogste bits naar de laagste vier. Zo niet dan is de vlag gereset.

6.6 SUBTRACT FLAG (N)

BCD-correctie is na optelling anders dan na aftrekking. De vlag is geset na aftrekking en gereset na optelling.

Instructie	C	Z	P/V	S	N	H	Commentaar
ADD A,s ADC A,s	↕	↕	V	↕	0	↕	8 bits optelling met of zonder carry
SUB s SBC A,s CP s NEG	↕	↕	V	↕	1	↕	8 bits aftrekking met of zonder carry, vergelijk en twee complement
AND s	0	↕	P	↕	0	1	logische operaties
OR s XOR s	0	↕	P	↕	0	0	
INC m	•	↕	V	↕	0	↕	8 bits increment
DEC m	•	↕	V	↕	1	↕	8 bits decrement
ADD DD,ss	↕	•	•	•	0	↕	16 bits optelling
ADC HL,ss	↕	↕	V	↕	0	↕	16 bits optelling met carry
SBC HL,ss	↕	↕	V	↕	1	↕	16 bits aftrekking met carry
RLA RLCA RRA RRCA	↕	•	•	•	0	0	roteer accumulator
RL m RLC m RR m RRC m SLA m SRA m SRL m	↕	•	P	↕	0	0	roteer en verschuif inhoud locatie m
RLD RRD	•	↕	P	↕	0	0	BCD rotatie
DAA	↕	↕	P	↕	•	↕	BCD correctie
CPL	•	•	•	•	1	1	complement accumulator
SCF	•	•	•	•	0	1	set carry flag
CCF	↕	•	•	•	0	↕	complement carry flag
IN r,(C)	•	↕	P	↕	0	0	input register indirect
INI IND OUTI OUTD	•	↕	X	↕	1	X	block input en output;
INIR INDR OTIR OTDR	•	1	X	1	1	X	Z=1 als B=0 anders Z=0
LDI LDD	•	•	↕	•	0	0	blok verplaats;
LDIR LDDR	•	•	0	•	0	0	P/V=0 als BC=0 anders P/V=1
CPI CPIR CPD CPDR	•	↕	↕	↕	1	↕	blok zoek; Z=1 als A=(HL) anders Z=0 P/V=0 als BC=0 anders P/V=1
LD A,I LD A,R	•	↕	F	↕	0	0	inhoud IFF2 in P/V
BIT b,s	•	↕	X	1	0	1	inhoud bit in Z

Betekenis van de gebruikte symbolen:
C=carry Z=zero P/V=parity/overflow S=sign N=substract H=half carry
↕ =afhankelijk van het resultaat van de instructie wordt de vlag geset of gereset
• =de vlag wordt niet veranderd door de instructie
0 =de vlag wordt gereset door de instructie
1 =de vlag wordt geset door de instructie
X =de vlag is onbepaald
V =P/V vlag geeft de overflow van het resultaat van de instructie
P =P/V vlag geeft de pariteit van het resultaat van de instructie

Afb. 6.1. Overzicht van de instructies die de vlaggen beïnvloeden.

7. Interrupts en busrequest

Een interrupt-request is een aanvraag van een randapparaat om het hoofdprogramma te onderbreken en een interrupt-service-routine af te handelen. De Z80 heeft twee interrupt-aansluitingen: één voor maskeerbare interrupts ($\overline{INT}$) en één voor niet-maskeerbare interrupts ($\overline{NMI}$).

7.1 MASKEERBARE INTERRUPTS

Deze interrupt kan worden verboden of toegestaan door de programmeur. Hiervoor zijn de instructies DI (Disable Interrupt) en EI (Enable Interrupt). In de Z80 bevinden zich twee interrupt-flipflops, IFF1 en IFF2. De instructie EI set de beide flipflops, DI reset ze.

Aan het eind van elke instructie wordt de $\overline{INT}$-aansluiting getest. Is er een interrupt-aanvraag en is IFF1 door een EI-instructie geset dan wordt de aanvraag behandeld. Is IFF1 gereset dan wordt de aanvraag genegeerd. Bij het accepteren van een maskeerbare interrupt worden IFF1 en IFF2 gereset om verdere interrupts te verbieden. De toestand van IFF2 kan worden getest met LD A,I of LD A,R. Beide instructies kopiëren IFF2 in de P/V-vlag.

De EI-instructie staat niet onmiddellijk interrupts toe maar pas aan het einde van de volgende instructie. Bij terugkeer uit een interrupt-subroutine moeten interrupts weer worden toegestaan. De opeenvolgende instructies EI en RETI (RETurn van Interrupt) zouden anders een interrupt tijdens een interrupt-routine toestaan. Nu worden interrupts pas weer toegestaan na de uitvoering van RETI, als is teruggesprongen naar het hoofdprogramma.

Bij blokvergelijking en blokverplaatsing wordt met het testen van de $\overline{INT}$-aansluiting niet gewacht tot de instructie is uitgevoerd. De test vindt plaats na elke afzonderlijke verplaatsing of vergelijking. Is er een interrupt-aanvraag dan wordt de uitvoering van de instructie stopgezet, de interrupt afgehandeld

en na RETI wordt de uitvoering van de instructie voortgezet.

7.2 INTERRUPT-MODES

Als maskeerbare interrupts zijn toegestaan kan de Z80 op drie manieren op een interrupt-aanvraag reageren. Keuze uit de drie interrupt-modes wordt gemaakt met de instructies IM 0 (Interrupt Mode 0), IM 1 en IM 2.
Na een reset staat de Z80 in interrupt-mode 0.

7.2.1 Interrupt-mode 0

Na een interrupt-aanvraag wacht de Z80 tot het randapparaat een instructie op de databus plaatst. Deze instructie wordt uitgevoerd. Een CALL of RST geeft een sprong naar een subroutine die met RETI moet worden afgesloten.

7.2.2 Interrupt-mode 1

In deze mode reageert de Z80 op een interrupt met een RST 0038h. Dat wil zeggen, de inhoud van de PC wordt op de stapel gezet en de PC wordt geladen met 0038h. Op dit adres moet dan of een subroutine beginnen of een sprong naar een subroutine staan.

7.2.3 Interrupt-mode 2

Na een interrupt-aanvraag wacht de Z80 tot het randapparaat een byte op de databus plaatst. Dit byte wordt opgevat als het lagere deel van een adres. Het hogere deel is de inhoud van het interrupt-pagina-adres-register I. Op dit adres en het volgende moet het startadres van een subroutine staan. De Z80 voert een CALL uit naar dat startadres: de inhoud van PC gaat naar de stapel en PC wordt geladen met het startadres. Op deze manier kan een randapparaat zelf naar de verlangde service-routine wijzen. Het 256 byte grote geheugenblok waarvan het hogere adresdeel overeenkomt met de inhoud van I, kan worden gebruikt als een interrupt-vectortabel waarvan de inhoud verwijst naar de startadressen van interrupt-service-routines.
Organisatie van de Z80 peripheral chips als de CTC en de PIO vereist dat van het byte, afkomstig van het randapparaat, het laagste bit nul is. De startadressen in de tabel beginnen dus allemaal op een even geheugenlocatie.
Een reset laadt het register I met nul.

7.3 NIET MASKEERBARE INTERRUPT

Een aanvraag voor een niet maskeerbare interrupt op de $\overline{\text{NMI}}$-aansluiting reset een interne flipflop. De toestand hiervan wordt aan het einde van elke instructie getest. De aanvraag wordt altijd behandeld. IFF1 wordt gereset om maskeerbare interrupts te negeren en de Z80 voert een restart naar adres 0066h uit. De inhoud van de PC gaat naar de stapel en de PC wordt geladen met 0066h. Op dat adres begint een subroutine of staat een sprong naar een subroutine.
Terugkeer uit een niet maskeerbare interrupt gebeurt met RETN (RETurn van Niet maskeerbare interrupt). Daarbij wordt IFF2 in IFF1 gekopieerd zodat wat de maskeerbare interrupts betreft de toestand gelijk is aan die van voor de niet maskeerbare interrupt. Aangezien een niet maskeerbare interrupt niet kan worden geweigerd, kunnen deze interrupts genest voorkomen. De niet maskeerbare interrupt heeft een hogere prioriteit dan de maskeerbare. Dat wil zeggen dat een niet maskeerbare interrupt altijd voor gaat en ook een service-routine van een maskeerbare interrupt zal onderbreken.
Tijdens de blokverplaatsingen en blokvergelijkingen wordt de toestand van de $\overline{\text{NMI}}$-aansluiting bekeken na elke afzonderlijke verplaatsing of vergelijking.

7.4 BUSREQUEST

De Z80 heeft een aparte busrequest-aansluiting: $\overline{\text{BUSRQ}}$. Aan het einde van elke machinecyclus wordt de aansluiting getest. Is er een busrequest dan stopt de Z80 met het uitvoeren van de instructie en zet de databus, adresbus en alle tristate controlbussen in toestand van hoge impedantie. De Z80 blijft in deze toestand tot het busrequest-signaal ten einde is. In de tussentijd kunnen adres-, data- en controlbussen worden gebruikt voor DMA (Direct Memory Access): een snelle data-uitwisseling tussen I/O en geheugen. Na de busrequest gaat de Z80 door met het uitvoeren van de onderbroken instructie.
Een busrequest heeft een hogere prioriteit dan niet maskeerbare en maskeerbare interrupts. Tijdens de busrequest worden interrupts genegeerd.
De Z80 kan tijdens een busrequest geen refreshes uitvoeren voor dynamische RAM.

8. Instructieset

Bij het beschrijven van de instructieset wordt de volgende notatie gebruikt:

- b wijst een enkel bit aan en heeft een waarde van 0 t/m 7
- cc specificeert de toestand van de vlaggen bij de voorwaardelijke instructies JR, JP, CALL en RET
- d een enkel byte met een waarde van -128 t/m +127
- e een enkel byte met een waarde van -126 t/m +129
- (HL) de inhoud van de geheugenlocatie waarnaar de inhoud van HL wijst
- m specificeert r, (HL), (IX+d) of (IY+d)
- n een enkel byte met een waarde van 0 t/m 255
- nn twee bytes met een waarde van 0 t/m 65535
- (nn) de inhoud van de geheugenlocatie waarnaar nn wijst
- pp één van de registerparen BC, DE, IX of SP
- qq één van de registerparen BC, DE, HL of AF
- r één van de registers A, B, C, D, E, H of L
- rr één van de registerparen BC, DE, HL of SP

- s specificeert r, n, (HL), (IX+d) of (IY+d)
- ss één van de registerparen BC, DE, HL of SP

Na het totale aantal T-cycli is voor elke instructie tussen haakjes aangegeven hoeveel T-cycli elke M-cyclus in beslag neemt.

ADC A,s (add with carry)

Telt de inhoud van de accumulator, de te specificeren operand s en de carry op en zet het resultaat in de accumulator.
Operand s kan zijn: r, n, (HL), (IX+d), (IY+d)

r	1 0 0 0 1 < r >	r=A	1 1 1	8F
		r=B	0 0 0	88
		r=C	0 0 1	89
		r=D	0 1 0	8A
		r=E	0 1 1	8B
		r=H	1 0 0	8C
		r=L	1 0 1	8D
n	1 1 0 0 1 1 1 0			CE
	< - - -n- - - >			data
(HL)	1 0 0 0 1 1 1 0			8E
(IX+d)	1 1 0 1 1 1 0 1			DD
	1 0 0 0 1 1 1 0			8E
	< - - -d- - - >			verplaatsing (twee-complement)
(IY+d)	1 1 1 1 1 1 0 1			FD
	1 0 0 0 1 1 1 0			8E
	< - - -d- - - >			verplaatsing (twee-complement)

carry flag : geset door een carry van bit 7, anders gereset
zero flag : geset als resultaat 0 is, anders gereset
parity/overflow : geset door overflow, anders gereset
sign flag : geset als resultaat negatief is (twee-compl.), anders gereset
subtract flag : gereset
half carry flag : geset door een carry van bit 3 naar 4, anders gereset

s	M-cycli	T-cycli	adressering s
r	1	4	register
n	2	7(4,3)	onmiddellijk
(HL)	2	7(4,3)	register indirect
(IX+d)	5	19(4,4,3,5,3)	geïndexeerd
(IY+d)	5	19(4,4,3,5,3)	geïndexeerd

ADC HL,ss (add with carry)

Telt de inhoud van HL, die van het te specificeren registerpaar ss en de carry bij elkaar op en zet het resultaat in HL.

1 1 1 0 1 1 0 1				ED
0 1 s s 1 0 1 0	ss=BC	0 0	4A	
	ss=DE	0 1	5A	
	ss=HL	1 0	6A	
	ss=SP	1 1	7A	

carry flag : geset door carry van bit 15, anders gereset
zero flag : geset als resultaat 0 is, anders gereset
parity/overflow : geset door overflow, anders gereset
sign flag : geset als resultaat negatief is (twee-compl.), anders gereset
subtract flag : gereset
half carry flag : geset door carry van bit 11 naar 12, anders gereset

4 M-cycli, 15(4,4,4,3) T-cycli, ss is register-geadresseerd.

ADD A,s

Telt de inhoud van de accumulator en de te specificeren operand s bij elkaar op en zet het resultaat in de accumulator.
Operand s kan zijn: r, n, (HL), (IX+d) of (IY+d)

r	1 0 0 0 0 < r >	r=A	1 1 1	87
		r=B	0 0 0	80
		r=C	0 0 1	81
		r=D	0 1 0	82
		r=E	0 1 1	83
		r=H	1 0 0	84
		r=L	1 0 1	85
n	1 1 0 0 0 1 1 0			C6
	< - - -n- - - >			data
(HL)	1 0 0 0 0 1 1 0			86
(IX+d)	1 1 0 1 1 1 0 1			DD
	1 0 0 0 0 1 1 0			86
	< - - -d- - - >			verplaatsing (twee-complement)
(IY+d)	1 1 1 1 1 1 0 1			FD
	1 0 0 0 0 1 1 0			86
	< - - -d- - - >			verplaatsing (twee-complement)

carry flag : geset door een carry van bit 7, anders gereset
zero flag : geset als resultaat 0 is, anders gereset
parity/overflow : geset door overflow, anders gereset
sign flag : geset als resultaat negatief is (twee-compl.), anders gereset
subtract flag : gereset
half carry flag : geset door een carry van bit 3 naar 4, anders gereset

s	M-cycli	T-cycli	adressering s
r	1	4	register
n	2	7(4,3)	onmiddellijk
(HL)	2	7(4,3)	register indirect
(IX+d)	5	19(4,4,3,5,3)	geïndexeerd
(IY+d)	5	19(4,4,3,5,3)	geïndexeerd

ADD HL,ss

Telt de inhoud van HL op bij die van het te specificeren registerpaar ss en zet het resultaat in HL.
ss is één van de registerparen BC, DE, HL of SP

0 0 s s 1 0 0 1	ss=BC	0 0	09
	ss=DE	0 1	19
	ss=HL	1 0	29
	ss=SP	1 1	39

carry flag : geset door carry van bit 15, anders gereset
zero flag : niet beïnvloed
parity/overflow : niet beïnvloed
sign flag : niet beïnvloed
subtract flag : gereset
half carry flag : geset door carry van bit 11 naar 12, anders gereset

3 M-cycli, 11(4,4,3) T-cycli. ss is register-geadresseerd.

ADD IX,pp

Telt de inhoud van IX op bij die van het te specificeren registerpaar pp en zet het resultaat in IX.
pp is één van de registerparen BC, DE, IX of SP

1 1 0 1 1 1 0 1	DD		
0 0 p p 1 0 0 1	pp=BC	0 0	09
	pp=DE	0 1	19
	pp=IX	1 0	29
	pp=SP	1 1	39

carry flag : geset door carry van bit 15, anders gereset
zero flag : niet beïnvloed
parity/overflow : niet beïnvloed
sign flag : niet beïnvloed
subtract flag : gereset
half carry flag : geset door carry van bit 11 naar 12, anders gereset

4 M-cycli, 15(4,4,4,3) T-cycli. pp is register-geadresseerd.

ADD IY,rr

Telt de inhoud van IY op bij die van het te specificeren registerpaar rr en zet het resultaat in IY.

rr is één van de registerparen BC, DE, IY of SP

1 1 1 1 1 1 0 1	FD		
0 0 r r 1 0 0 1	rr=BC	0 0	09
	rr=DE	0 1	19
	rr=IY	1 0	29
	rr=SP	1 1	39

carry flag : geset door carry van bit 15, anders gereset
zero flag : niet beïnvloed
parity/overflow : niet beïnvloed
sign flag : niet beïnvloed
subtract flag : gereset
half carry flag : geset door carry van bit 11 naar 12, anders gereset

4 M-cycli, 15(4,4,4,3) T-cycli. rr is register-geadresseerd.

AND s

Logische EN-operatie van de inhoud van de accumulator en de te specificeren operand s. Het resultaat komt in de accumulator.

Operand s kan zijn: r, n, (HL), (IX+d) of (IY+d)

r	1 0 1 0 0 < r >	r=A	1 1 1	A7
		r=B	0 0 0	A0
		r=C	0 0 1	A1
		r=D	0 1 0	A2
		r=E	0 1 1	A3
		r=H	1 0 0	A4
		r=L	1 0 1	A5
n	1 1 1 0 0 1 1 0	E6		
	< - - -n- - - >	data		
(HL)	1 0 1 0 0 1 1 0	A6		
(IX+d)	1 1 0 1 1 1 0 1	DD		
	1 0 1 0 0 1 1 0	A6		
	< - - -d- - - >	verplaatsing (twee-complement)		
(IY+d)	1 1 1 1 1 1 0 1	FD		
	1 0 1 0 0 1 1 0	A6		
	< - - -d- - - >	verplaatsing (twee-complement)		

carry flag : gereset
zero flag : geset als resultaat nul is, anders gereset
parity/overflow : geset door even pariteit, anders gereset
sign flag : geset als resultaat negatief is (twee-compl.), anders gereset
subtract flag : gereset
half carry flag : geset

s	M-cycli	T-cycli	adressering s
r	1	4	register
n	2	7(4,3)	onmiddellijk
(HL)	2	7(4,3)	register indirect
(IX+d)	5	19(4,4,3,5,3)	geïndexeerd
(IY+d)	5	19(4,4,3,5,3)	geïndexeerd

BIT b,(HL)

Test bit b van de inhoud van de door HL aangewezen geheugenlocatie. Het resultaat (de inverse van b) komt in de zero flag.

```
1 1 0 0 1 0 1 1   CB
0 1 < b > 1 1 0   b=bit 0   0 0 0   46
                  b=bit 1   0 0 1   4E
                  b=bit 2   0 1 0   56
                  b=bit 3   0 1 1   5E
                  b=bit 4   1 0 0   66
                  b=bit 5   1 0 1   6E
                  b=bit 6   1 1 0   76
                  b=bit 7   1 1 1   7E
```

carry flag : niet beïnvloed
zero flag : geset als het bit 0 is, anders gereset
parity/overflow : ongedefinieerd
sign flag : ongedefinieerd
subtract flag : gereset
half carry flag : geset

3 M-cycli, 12(4,4,4) T-cycli. b is bit-geadresseerd, HL register indirect.

BIT b,(IX+d)

Test bit b van de inhoud van de door (IX+d) aangewezen geheugenlocatie. Het resultaat (de inverse van b) komt in de zero flag.

```
1 1 0 1 1 1 0 1   DD
1 1 0 0 1 0 1 1   CB
< - - -d- - - >   verplaatsing (twee-complement)
0 1 < b > 1 1 0   b=bit 0   0 0 0   46
                  b=bit 1   0 0 1   4E
                  b=bit 2   0 1 0   56
                  b=bit 3   0 1 1   5E
                  b=bit 4   1 0 0   66
                  b=bit 5   1 0 1   6E
                  b=bit 6   1 1 0   76
                  b=bit 7   1 1 1   7E
```


BIT b,(IX+d) (vervolg)

carry flag : niet beïnvloed
zero flag : geset als het bit 0 is, anders gereset
parity/overflow : ongedefinieerd
sign flag : ongedefinieerd
subtract flag : gereset
half carry flag : geset

5 M-cycli, 20(4,4,3,5,4) T-cycli. b is bit-geadresseerd, (IX+d) geïndexeerd.

BIT b,(IY+d)

Test bit b van de inhoud van de door (IY+d) aangewezen geheugenlocatie. Het resultaat (de inverse van b) komt in de zero flag.

```
1 1 1 1 1 1 0 1   FD
1 1 0 0 1 0 1 1   CB
< - - -d- - - >   verplaatsing (twee-complement)
0 1 < b > 1 1 0   b=bit 0   0 0 0   46
                  b=bit 1   0 0 1   4E
                  b=bit 2   0 1 0   56
                  b=bit 3   0 1 1   5E
                  b=bit 4   1 0 0   66
                  b=bit 5   1 0 1   6E
                  b=bit 6   1 1 0   76
                  b=bit 7   1 1 1   7E
```

carry flag : niet beïnvloed
zero flag : geset als het bit 0 is, anders gereset
parity/overflow : ongedefinieerd
sign flag : ongedefinieerd
subtract flag : gereset
half carry flag : geset

5 M-cycli, 20(4,4,3,5,4) T-cycli. b is bit-geadresseerd, (IY+d) geïndexeerd.

BIT b,r

Test bit b van de inhoud van het door r gespecificeerde register. Het resultaat (de inverse van b) komt in de zero flag.

```
1 1 0 0 1 0 1 1   CB
0 1 < b > < r >   b=bit 0   0 0 0   r=A   1 1 1
                  b=bit 1   0 0 1   r=B   0 0 0
                  b=bit 2   0 1 0   r=C   0 0 1
                  b=bit 3   0 1 1   r=D   0 1 0
                  b=bit 4   1 0 0   r=E   0 1 1
                  b=bit 5   1 0 1   r=H   1 0 0
                  b=bit 6   1 1 0   r=L   1 0 1
                  b=bit 7   1 1 1
```

	A	B	C	D	E	H	L
0	47	40	41	42	43	44	45
1	4F	48	49	4A	4B	4C	4D
2	57	50	51	52	53	54	55
3	5F	58	59	5A	5B	5C	5D
4	67	60	61	62	63	64	65
5	6F	68	69	6A	6B	6C	6D
6	77	70	71	72	73	74	75
7	7F	78	79	7A	7B	7C	7D

carry flag : niet beïnvloed
zero flag : geset als het bit 0 is, anders gereset
parity/overflow : ongedefinieerd
sign flag : ongedefinieerd
subtract flag : gereset
half carry flag : geset

2 M-cycli, 8(4,4) T-cycli. b is bit-geadresseerd, r is register-geadresseerd.

CALL cc,nn

Als de in cc gestelde voorwaarde juist is, wordt de PC op de stapel gezet en adres nn in de PC geladen. In het programma wordt gesprongen naar een subroutine die op adres nn begint. Bij terugkeer uit de subroutine (RET of RET cc) worden de twee bytes op de top van de stapel in de PC geladen. Is de gestelde voorwaarde niet juist dan wordt de volgende instructie uitgevoerd.

cc kan zijn: NZ, Z, NC, C, PO, PE, P of M

```
1 1 <c  c> 1 0 0
< - - -n- - - >   lagere orde deel-adres
< - - -n- - - >   hogere orde deel-adres
```

cc=NZ	(non zero)	0 0 0	C4
cc=Z	(zero)	0 0 1	CC
cc=NC	(non carry)	0 1 0	D4
cc=C	(carry)	0 1 1	DC
cc=PO	(pariteit oneven)	1 0 0	E4
cc=PE	(pariteit even)	1 0 1	EC
cc=P	(plus)	1 1 0	F4
cc=M	(min)	1 1 1	FC

Vlaggen niet beïnvloed.

Als cc juist is: 5 M-cycli, 17(4,3,4,3,3) T-cycli.
Is cc onjuist: 3 M-cycli, 10(4,3,3) T-cycli. nn is uitgebreid geadresseerd.

CALL nn

Zet de inhoud van PC op de stapel en laadt adres nn in de PC. In het programma wordt gesprongen naar een subroutine die begint op adres nn. Terugkeer uit de subroutine (RET of RET cc) zet de twee bytes op de top van de stapel in de PC.

```
1 1 0 0 1 1 0 1   CD
< - - -n- - - >   lagere orde deel-adres
< - - -n- - - >   hogere orde deel-adres
```

Vlaggen niet beïnvloed.

5 M-cycli, 17(4,3,4,3,3) T-cycli. nn is uitgebreid geadresseerd.

CCF (complement carry flag)

Complementeert het carry flag bit.

```
0 0 1 1 1 1 1 1   3F
```

carry flag : geset als de carry 0 was, anders gereset
zero flag : onbeïnvloed
parity/overflow : onbeïnvloed
sign flag : onbeïnvloed
subtract flag : gereset
half carry flag : kopie van de oorspronkelijke carry

1 M-cyclus, 4 T-cycli. De carry flag is impliciet geadresseerd.

CP s (compare)

Trekt de door s te specificeren operand af van de inhoud van de accumulator. Het resultaat wordt genegeerd maar beïnvloedt wel de vlaggen.

s kan zijn: r, n, (HL), (IX+d) of (IY+d)

```
r          1 0 1 1 1 < r >   r=A  1 1 1   BF
                             r=B  0 0 0   B8
                             r=C  0 0 1   B9
```


CP s (vervolg)

```
                             r=D  0 1 0   BA
                             r=E  0 1 1   BB
                             r=H  1 0 0   BC
                             r=L  1 0 1   BD

n          1 1 1 1 1 1 1 0   FE
           < - - -n- - - >   data

(HL)       1 0 1 1 1 1 1 0   BE

(IX+d)     1 1 0 1 1 1 0 1   DD
           1 0 1 1 1 1 1 0   BE
           < - - -d- - - >   verplaatsing (twee-complement)

(IY+d)     1 1 1 1 1 1 0 1   FD
           1 0 1 1 1 1 1 0   BE
           < - - -d- - - >   verplaatsing (twee-complement)
```

carry flag : geset bij een borrow van bit 7, anders gereset
zero flag : geset als A=s, anders gereset
parity/overflow : geset door overflow, anders gereset
sign flag : geset als het resultaat negatief is, anders gereset
subtract flag : geset
half carry flag : geset door borrow van bit 4 naar 3, anders gereset

s	M-cycli	T-cycli	adresseringswijze
r	1	4	register
n	2	7(4,3)	onmiddellijk
(HL)	2	7(4,3)	register indirect
(IX+d)	5	19(4,4,3,5,3)	geïndexeerd
(IY+d)	5	19(4,4,3,5,3)	geïndexeerd

CPD (compare decrement)

Trekt de inhoud van de geheugenlocatie waar HL naar wijst af van de accumulator. Het resultaat wordt genegeerd maar beïnvloedt wel de vlaggen. Vervolgens worden de inhouden van HL en (teller) BC met 1 verlaagd.

```
1 1 1 0 1 1 0 1   ED
1 0 1 0 1 0 0 1   A9
```

carry flag : onbeïnvloed
zero flag : geset als A=(HL), anders gereset
parity/overflow : gereset als BC=0 na de instructie, anders geset
sign flag : geset als het resultaat negatief is, anders gereset
subtract flag : geset
half carry flag : geset door een borrow van bit 4 naar 3, anders gereset

4 M-cycli, 16(4,4,3,5) T-cycli.

CPDR (compare decrement repeat)

Trekt de inhoud van de geheugenlocatie waar HL naar wijst af van de accumulator. Het resultaat wordt genegeerd maar beïnvloedt wel de vlaggen. Vervolgens worden de inhouden van HL en (teller) BC met 1 verlaagd. De instructie wordt, door de inhoud van PC met 2 te verminderen, herhaald tot teller BC=0 of A=(HL).

```
1 1 1 0 1 1 0 1   ED
1 0 1 1 1 0 0 1   B9
```

carry flag : onbeïnvloed
zero flag : geset als A=(HL), anders gereset
parity/overflow : gereset als BC=0 na de instructie, anders geset
sign flag : geset als het resultaat negatief is, anders gereset
subtract flag : geset
half carry flag : geset door borrow van bit 4 naar 3, anders gereset

CPDR (vervolg)

Voor BC=0 of A=(HL): 4 M-cycli, 16(4,4,3,5) T-cycli (laatste vergelijking).
Voor BC ongelijk 0 en A ongelijk (HL): 5 M-cycli, 21(4,4,3,5,5) T-cycli (elke andere vergelijking dan de laatste).

CPI (compare increment)

Trekt de inhoud van de geheugenlocatie waar HL naar wijst af van de accumulator. Het resultaat wordt genegeerd maar beïnvloedt wel de vlaggen. Vervolgens wordt de inhoud van HL met 1 verhoogd en die van (teller) BC met 1 verlaagd.

```
1 1 1 0 1 1 0 1   ED
1 0 1 0 0 0 0 1   A1
```

carry flag : onbeïnvloed
zero flag : geset als A=(HL), anders gereset
parity/overflow : gereset als BC=0 na de instructie, anders geset
sign flag : geset als het resultaat negatief is, anders gereset
subtract flag : geset
half carry flag : geset door borrow van bit 4 naar 3, anders gereset

4 M-cycli, 16(4,4,3,5) T-cycli.

CPIR (compare increment repeat)

Trekt de inhoud van de geheugenlocatie waar HL naar wijst af van de accumulator. Het resultaat wordt genegeerd maar beïnvloedt wel de vlaggen. Vervolgens wordt de inhoud van HL met 1 verhoogd, die van teller BC met 1 verlaagd. Behalve wanneer A=(HL) of BC=0 wordt de instructie herhaald door de inhoud van PC met twee te verminderen.

1 1 1 0 1 1 0 1 ED
1 0 1 1 0 0 0 1 B1

carry flag : onbeïnvloed
zero flag : geset als A=(HL), anders gereset
parity/overflow : gereset als BC=0 na de instructie, anders geset
sign flag : geset als het resultaat negatief is, anders gereset
subtract flag : geset
half carry flag : geset door borrow van bit 4 naar 3, anders gereset

Voor BC=0 of A=(HL): 4 M-cycli, 16(4,4,3,5) T-cycli (laatste vergelijking). Is BC ongelijk aan 0 en A ongelijk aan (HL): 5 M-cycli, 21(4,4,3,5,5) T-cycli (elke andere vergelijking dan de laatste).

CPL (complement)

Inverteert de inhoud van de accumulator (één-complement).

0 0 1 0 1 1 1 1 2F

carry flag : onbeïnvloed
zero flag : onbeïnvloed
parity/overflow : onbeïnvloed
sign flag : onbeïnvloed
subtract flag : geset
half carry flag : geset

1 M-cyclus, 4 T-cycli. De accumulator is impliciet geadresseerd.

DAA (decimal adjust accumulator)

Corrigeert na optelling (ADD, ADC, INC) of aftrekking (SBC, SUB, DEC, NEG) de inhoud van de accumulator voor packed BCD.

0 0 1 0 0 1 1 1 27

De correcties zijn:

voorafgaande instructie	carry voor DAA	inhoud acc. bit 7-4 (H)	half carry voor DAA	inhoud acc. bit 3-0 (H)	opgeteld bij inhoud acc.	carry na DAA
ADD	0	0-9	0	0-9	00	0
ADC	0	0-8	0	A-F	06	0
INC	0	0-9	1	0-3	06	0
	0	A-F	0	0-9	60	1
	0	9-F	0	A-F	66	1
	0	A-F	1	0-3	66	1
	1	0-2	0	0-9	60	1
	1	0-2	0	A-F	66	1
	1	0-3	1	0-3	66	1
SUB	0	0-9	0	0-9	00	0
SBC	0	0-8	1	6-F	FA	0
DEC	1	7-F	0	0-9	A0	1
NEG	1	6-F	1	6-F	9A	1

carry flag : geset bij een carry of borrow voor decimale getallen; carry geeft aan dat het resultaat groter is dan 99
zero flag : geset als het resultaat 0 is, anders gereset
parity/overflow : geset door even pariteit, anders gereset
sign flag : kopie van bit 7 van accumulator
subtract flag : onbeïnvloed
half carry flag : geset bij een carry of borrow van of naar bit 3 voor decimale getallen

1 M-cyclus, 4 T-cycli. De accumulator is impliciet geadresseerd.

DEC IX (decrement)

Vermindert de inhoud van index-register IX met 1.

```
1 1 0 1 1 1 0 1   DD
0 0 1 0 1 0 1 1   2B
```

Vlaggen niet beïnvloed.

2 M-cycli, 10(4,6) T-cycli.

DEC IY (decrement)

Vermindert de inhoud van index-register IY met 1.

```
1 1 1 1 1 1 0 1   FD
0 0 1 0 1 0 1 1   2B
```

Vlaggen niet beïnvloed.

DEC m (decrement)

Vermindert de door m te specificeren operand met 1.

m kan zijn: r, (HL), (IX+d) of (IY+d)

```
r          0 0 < r > 1 0 1    r=A  1 1 1   3D
                              r=B  0 0 0   05
                              r=C  0 0 1   0D
                              r=D  0 1 0   15
                              r=E  0 1 1   1D
                              r=H  1 0 0   25
                              r=L  1 0 1   2D

(HL)       0 0 1 1 0 1 0 1    35

(IX+d)     1 1 0 1 1 1 0 1    DD
           0 0 1 1 0 1 0 1    35
           < - - -d- - - >    verplaatsing (twee-complement)

(IY+d)     1 1 1 1 1 1 0 1    FD
           0 0 1 1 0 1 0 1    35
           < - - -d- - - >    verplaatsing (twee-complement)
```

carry flag : onbeïnvloed
zero flag : geset als het resultaat 0 is, anders gereset
parity/overflow : geset als m 80H was voor operatie en dus van teken verandert, anders gereset
sign flag : geset als het resultaat negatief is, anders gereset
subtract flag : geset
half carry flag : geset bij een borrow van bit 4, anders gereset

	M-cycli	T-cycli	adresseringswijze m
r	1	4	register
(HL)	3	11(4,4,3)	register indirect
(IX+d)	6	23(4,4,3,5,4,3)	geïndexeerd
(IY+d)	6	23(4,4,3,5,4,3)	geïndexeerd

DEC ss (decrement)

Vermindert de inhoud van het door ss te specificeren register-paar met 1.

ss kan zijn: BC, DE, HL of SP

0 0 s s 1 0 1 1	ss=BC	0 0 0B
	ss=DE	0 1 1B
	ss=HL	1 0 2B
	ss=SP	1 1 3B

Vlaggen niet beïnvloed.

1 M-cyclus, 6 T-cycli.

DI (disable interrupts)

Reset de interrupt-flipflops IFF1 en IFF2 waardoor maskeerbare interrupts niet meer worden geaccepteerd. Maskeerbare interrupts worden weer toegestaan na EI.

1 1 1 1 0 0 1 1 F3

Vlaggen niet beïnvloed.

1 M-cyclus, 4 T-cycli. Beide flipflops zijn impliciet geadres-seerd.

DJNZ e (decrement, jump if non zero)

Vermindert de inhoud van register B met 1. Is B hierna niet 0 dan wordt de verplaatsing opgeteld bij de inhoud van PC. Deze wijst dan al naar de volgende instructie en is dus al 2 hoger dan aan het begin van DJNZ e. Hoewel de inhoud van het verplaat-sings-byte +127 t/m -128 is, kan ten opzichte van het begin van DJNZ e van +129 t/m -126 worden gesprongen.

0 0 0 1 0 0 0 0	10
< - e min 2 - >	verplaatsing (twee-complement)

Vlaggen niet beïnvloed.

Als B nul is: 2 M-cycli, 8(5,3) T-cycli
Als B niet nul is: 3 M-cycli, 13(5,3,5) T-cycli
Relatieve adressering.

EI (enable interrupt)

Set de interrupt-flipflops IFF1 en IFF2 zodat maskeerbare interrupts zijn toegestaan na uitvoering van de instructie volgend op EI (meestal RETI).

1 1 1 1 1 0 1 1 FB

Vlaggen niet beïnvloed.

1 M-cyclus, 4 T-cycli. De flipflops zijn impliciet geadresseerd.

EX AF,AF' (exchange)

Verwisselt de inhouden van de registerparen AF en AF'.

0 0 0 0 1 0 0 0 08

Vlaggen: F en F' worden verwisseld.

1 M-cyclus, 4 T-cycli.

EX DE,HL (exchange)

Verwisselt de inhoud van de registerparen DE en HL.

1 1 1 0 1 0 1 1 EB

Vlaggen niet beïnvloed.

1 M-cyclus, 4 T-cycli.

EX (SP),HL (exchange)

Verwisselt de inhoud van L met die van de geheugenlocatie waar SP naar wijst en de inhoud van H met die van de geheugenlocatie waarnaar SP+1 wijst.

1 1 1 0 0 0 1 1 E3

Vlaggen niet beïnvloed.

5 M-cycli, 19(4,3,4,3,5) T-cycli.

EX (SP),IX (exchange)

Verwisselt de inhoud van X met die van de geheugenplaats waar SP naar wijst en de inhoud van I met die van de geheugenlocatie waarnaar SP+1 wijst.

1 1 0 1 1 1 0 1 DD
1 1 1 0 0 0 1 1 E3

Vlaggen niet beïnvloed.

6 M-cycli, 23(4,4,3,4,3,5) T-cycli.

EX (SP),IY (exchange)

Verwisselt de inhoud van Y met die van de geheugenplaats waar SP naar wijst en de inhoud van I met die van de geheugenlocatie waarnaar SP+1 wijst.

1 1 1 1 1 1 0 1 FD
1 1 1 0 0 0 1 1 E3

Vlaggen niet beïnvloed.

6 M-cycli, 23(4,4,3,4,3,5) T-cycli.

EXX (exchange)

Verwisselt de inhouden van de registerparen BC, DE en HL met die van de alternatieve registerparen BC', DE' en HL'.

1 1 0 1 1 0 0 1 D9

Vlaggen niet beïnvloed.

1 M-cyclus, 4 T-cycli. Registerparen impliciet geadresseerd.

HALT

Stopt de uitvoering van het programma tot een maskeerbare interrupt (mits toegestaan), een niet maskeerbare interrupt of een reset wordt ontvangen. In de HALT-toestand voert de Z80 NOP-instructies uit.

0 1 1 1 0 1 1 0 76

Vlaggen niet beïnvloed.

1 M-cyclus, 4 T-cycli.

IM 0 (interrupt-mode 0)

Zet de Z80 in interrupt-mode 0. In deze mode moet het apparaat dat een maskeerbare interrupt (mits toegestaan) aanvraagt een instructie op de databus zetten. De Z80 voert deze instructie uit.

```
1 1 1 0 1 1 0 1   ED
0 1 0 0 0 1 1 0   46
```

Vlaggen niet beïnvloed.
2 M-cycli, 8(4,4) T-cycli.

IM 1 (interrupt-mode 1)

Zet de Z80 in interrupt-mode 1. In deze mode reageert de Z80 op een maskeerbare interrupt (mits toegestaan) aanvraag met het uitvoeren van een RST 38H-instructie.

```
1 1 1 0 1 1 0 1   ED
0 1 0 1 0 1 1 0   56
```

Vlaggen niet beïnvloed.
2 M-cycli, 8(4,4) T-cycli.

IM 2 (interrupt-mode 2)

Zet de Z80 in interrupt-mode 2. Het apparaat dat de maskeerbare interrupt (mits toegestaan) aanvraagt moet het lagere orde-deel van een adres, waarvan het laagste bit een 0 dient te zijn, op de databus zetten. Het hogere orde-deel is de inhoud van register I. Het adres wordt in de PC geladen. De Z80 zet de oorspronkelijke inhoud van de PC op de stapel.

```
1 1 1 0 1 1 0 1   ED
0 1 0 1 1 1 1 0   5E
```

Vlaggen niet beïnvloed.

2 M-cycli, 8(4,4) T-cycli.

IN A,(n) (input)

Laadt de accumulator vanuit input-poort n. De waarde van n komt op het lagere deel van de adresbus (A0 t/m A7) en de inhoud van de accumulator op het hogere deel (A8 t/m A15). De Z80 leest het door de geselecteerde poort op de databus geplaatste byte en zet het in de accumulator.

```
1 1 0 1 1 0 1 1   DB
< - - -n- - - >   poort
```

Vlaggen niet beïnvloed.
3 M-cycli, 11(4,3,4) T-cycli. De poort is direct geadresseerd.

IN r,(C) (input)

Laadt register r vanuit input-poort C. De inhoud van register C komt op het lagere deel van de adresbus (A0 t/m A7) en de inhoud van register B op het hogere deel (A8 t/m A15). De Z80 leest het door de geselecteerde poort op de databus geplaatste byte en zet het in het door r gespecificeerde register.

r kan zijn: A, B, C, D, E, H of L

```
1 1 1 0 1 1 0 1   ED
0 1 < r > 0 0 0   r=A   1 1 1   78
                  r=B   0 0 0   40
                  r=C   0 0 1   48
                  r=D   0 1 0   50
                  r=E   0 1 1   58
                  r=H   1 0 0   60
                  r=L   1 0 1   68
```

carry flag : niet beïnvloed
zero flag : geset als ingelezen data 0 is, anders gereset
parity/overflow : geset door even pariteit, anders gereset
sign flag : geset als ingelezen data negatief is, anders gereset
subtract flag : gereset
half carry flag : gereset

3 M-cycli, 12(4,4,4) T-cycli. Poortregister indirect geadresseerd.

INC IX (increment)

Verhoogt de inhoud van register IX met één.

```
1 1 0 1 1 1 0 1    DD
0 0 1 0 0 0 1 1    23
```

Vlaggen niet beïnvloed.

2 M-cycli, 10(4,6) T-cycli.

INC IY (increment)

Verhoogt de inhoud van register IY met één.

```
1 1 1 1 1 1 0 1    FD
0 0 1 0 0 0 1 1    23
```

Vlaggen niet beïnvloed.

2 M-cycli, 10(4,6) T-cycli.

INC m (increment)

Verhoogt de door m te specificeren operand met één.

m kan zijn: r, (HL), (IX+d) of (IY+d)

```
r          0 0 < r > 1 0 0     r=A  1 1 1   3C
                               r=B  0 0 0   04
                               r=C  0 0 1   0C
                               r=D  0 1 0   14
                               r=E  0 1 1   1C
                               r=H  1 0 0   24
                               r=L  1 0 1   2C

(HL)       0 0 1 1 0 1 0 0     34

(IX+d)     1 1 0 1 1 1 0 1     DD
           0 0 1 1 0 1 0 0     34
           < - - -d- - - >     verplaatsing (twee-complement)

(IY+d)     1 1 1 1 1 1 0 1     FD
           0 0 1 1 0 1 0 0     34
           < - - -d- - - >     verplaatsing (twee-complement)
```

carry flag : niet beïnvloed
zero flag : geset als het resultaat 0 is, anders gereset
parity/overflow : geset als m=7FH voor de operatie en dus van teken verandert, anders gereset
sign flag : geset als het resultaat negatief is, anders gereset
subtract flag : gereset
half carry flag : geset door een carry van bit 3 naar 4

m	M-cycli	T-cycli	adressering
r	1	4	register
(HL)	3	11(4,4,3)	register indirect
(IX+d)	6	23(4,4,3,5,4,3)	geïndexeerd
(IY+d)	6	23(4,4,3,5,4,3)	geïndexeerd

INC ss

Verhoogt de inhoud van het door ss gespecificeerde registerpaar met één.

ss kan zijn: BC, DE, HL of SP

0 0 s s 0 0 1 1	ss=BC	0 0	03
	ss=DE	0 1	13
	ss=HL	1 0	23
	ss=SP	1 1	33

Vlaggen niet beïnvloed.

1 M-cyclus, 6 T-cycli. ss is register-geadresseerd.

IND (input decrement)

Leest de door de inhoud van register C aangewezen poort en zet het resultaat in de geheugenplaats waarnaar de inhoud van registerpaar HL wijst. De Z80 zet daartoe de inhoud van register C op het lagere orde-deel van de adresbus (A0 t/m A7) en de inhoud van register B op het hogere orde-deel (A8 t/m A15). Aan het eind van de instructie wordt de inhoud van het als teller te gebruiken register B en die van het registerpaar HL met één verminderd.

1 1 1 0 1 1 0 1	ED
1 0 1 0 1 0 1 0	AA

carry flag	: niet beïnvloed
zero flag	: geset als teller B=0, anders gereset
parity/overflow	: ongedefinieerd
sign flag	: ongedefinieerd
subtract flag	: geset
half carry flag	: ongedefinieerd

4 M-cycli, 16(4,5,3,4) T-cycli.

INDR (input decrement repeat)

Leest de door de inhoud van register C aangewezen poort en zet het resultaat in de geheugenplaats waarnaar de inhoud van registerpaar HL wijst. De Z80 zet daartoe de inhoud van register C op het lagere orde-deel van de adresbus (A0 t/m A7) en de inhoud van register B op het hogere orde-deel (A8 t/m A15). Aan het eind van de instructie wordt de inhoud van registerpaar HL en die van het als teller te gebruiken register B met één verminderd. De Z80 herhaalt de instructie, door de inhoud van de PC met twee te verminderen, totdat de inhoud van register B gelijk is aan nul.

1 1 1 0 1 1 0 1	ED
1 0 1 1 1 0 1 0	BA

carry flag	: niet beïnvloed
zero flag	: geset
parity/overflow	: ongedefinieerd
sign flag	: ongedefinieerd
subtract flag	: geset
half carry flag	: ongedefinieerd

Als B gelijk is aan nul: 4 M-cycli, 16(4,5,3,4) T-cycli
Als B ongelijk is aan nul: 5 M-cycli, 21(4,5,3,4,5) T-cycli

INI (input increment)

Leest de door de inhoud van register C aangewezen poort en zet het resultaat in de geheugenplaats waarnaar de inhoud van registerpaar HL wijst. De Z80 zet daartoe de inhoud van register C op het lagere orde-deel van de adresbus (A0 t/m A7) en de inhoud van register B op het hogere orde-deel (A8 t/m A15). Aan het eind van de instructie wordt de inhoud van het als teller te gebruiken register B met één verminderd en die van registerpaar HL met één vermeerderd.

1 1 1 0 1 1 0 1	ED
1 0 1 0 0 0 1 0	A2

INI (vervolg)

carry flag : niet beïnvloed
zero flag : geset als teller B=0, anders gereset
parity/overflow : ongedefinieerd
sign flag : ongedefinieerd
subtract flag : geset
half carry flag : ongedefinieerd

4 M-cycli, 16(4,5,3,4) T-cycli

INIR (input increment repeat)

Leest de door de inhoud van register C aangewezen poort en zet het resultaat in de geheugenplaats waarnaar de inhoud van registerpaar HL wijst. De Z80 zet daartoe de inhoud van register C op het lagere orde-deel van de adresbus (A0 t/m A7) en de inhoud van register B op het hogere orde-deel (A8 t/m A15). Aan het eind van de instructie wordt de inhoud van registerpaar HL met één vermeerderd en die van het als teller te gebruiken register B met één verminderd. De Z80 herhaalt de instructie, door de inhoud van PC met twee te verminderen, totdat de inhoud van register B gelijk is aan nul.

1 1 1 0 1 1 0 1 ED
1 0 1 1 0 0 1 0 B2

carry flag : niet beïnvloed
zero flag : geset
parity/overflow : ongedefinieerd
sign flag : ongedefinieerd
subtract flag : geset
half carry flag : ongedefinieerd

Als B gelijk is aan nul: 4 M-cycli, 16(4,5,3,4) T-cycli
Als B ongelijk is aan nul: 5 M-cycli, 21(4,5,3,4,5) T-cycli

JP cc,nn (jump)

Laadt, als de door cc te specificeren conditie waar is, adres nn in de PC. Is de door cc te specificeren conditie niet waar dan gaat de Z80 verder met de volgende instructie.

1 1 <c c> 0 1 0

cc=NZ	non zero	0 0 0	C2	zero flag
cc=Z	zero	0 0 1	CA	zero flag
cc=NC	non carry	0 1 0	D2	carry
cc=C	carry	0 1 1	DA	carry
cc=PO	pariteit oneven	1 0 0	E2	p/o
cc=PE	pariteit even	1 0 1	EA	p/o
cc=P	plus	1 1 0	F2	sign flag
cc=M	min	1 1 1	FA	sign flag

< - - -n- - - > lagere orde-deel adres
< - - -n- - - > hogere orde-deel adres

Vlaggen onbeïnvloed.

3 M-cycli, 10(4,3,3) T-cycli.
Het sprongadres is uitgebreid geadresseerd.

JP (HL) (jump)

Laadt de PC met de inhoud van registerpaar HL.

1 1 1 0 1 0 0 1 E9

Vlaggen onbeïnvloed.

1 M-cyclus, 4 T-cycli.
Het sprongadres is register indirect geadresseerd.

JP (IX) (jump)

Laadt de PC met de inhoud van registerpaar IX.

```
1 1 0 1 1 1 0 1   DD
1 1 1 0 1 0 0 1   E9
```

Vlaggen onbeïnvloed.

2 M-cycli, 8(4,4) T-cycli.
Het sprongadres is register indirect geadresseerd.

JP (IY) (jump)

Laadt de PC met de inhoud van registerpaar IY.

```
1 1 1 1 1 1 0 1   FD
1 1 1 0 1 0 0 1   E9
```

Vlaggen onbeïnvloed.

2 M-cycli, 8(4,4) T-cycli.
Het sprongadres is register indirect geadresseerd.

JP nn (jump)

Laadt het door nn te specificeren adres in de PC.

```
1 1 0 0 0 0 1 1   C3
< - - -n- - - >   lagere orde-deel adres
< - - -n- - - >   hogere orde-deel adres
```

Vlaggen onbeïnvloed.

3 M-cycli, 10(4,3,3) T-cycli.
Het sprongadres is uitgebreid geadresseerd.

JR cc,e (jump relative)

Telt, als de door cc te specificeren conditie waar is, de verplaatsing op bij de momentele inhoud van de PC. Die is dan twee hoger dan aan het begin van JR cc,e. Als ten opzichte van het begin van de instructie de sprong e is, moet bij de PC e-2 worden opgeteld. Als de conditie niet waar is gaat de Z80 verder met de volgende instructie.

```
0 0 1 c c 0 0 0
     cc=NZ  non zero     0 0   20  zero flag
     cc=Z   zero         0 1   28  zero flag
     cc=NC  non carry    1 0   30  carry
     cc=C   carry        1 1   38  carry

<     e-2     >   verplaatsing (twee-complement)
```

Vlaggen onbeïnvloed.

Als de conditie niet waar is: 2 M-cycli, 7(4,3) T-cycli
Als de conditie waar is: 3 M-cycli, 12(4,3,5) T-cycli
Relatieve adressering.

JR e (jump relative)

Telt de verplaatsing op bij de momentele inhoud van de PC. Die is dan twee keer hoger dan aan het begin van JR e. Als ten opzichte van het begin van de instructie de sprong e is, moet bij de PC e-2 worden opgeteld.

```
0 0 0 1 1 0 0 0   18
<     e-2     >   verplaatsing (twee-complement)
```

Vlaggen onbeïnvloed.

3 M-cycli, 12(4,3,5) T-cycli.
Relatieve adressering.

LD A,(BC) (load)

Laadt de accumulator met de inhoud van de geheugenlocatie waarnaar de inhoud van registerpaar BC wijst.

0 0 0 0 1 0 1 0 0A

Vlaggen onbeïnvloed.

2 M-cycli, 7(4,3) T-cycli. De bron is register indirect geadresseerd.

LD A,(DE) (load)

Laadt de accumulator A met de inhoud van de geheugenlocatie waarnaar de inhoud van registerpaar DE wijst.

0 0 0 1 1 0 1 0 1A

Vlaggen onbeïnvloed.

2 M-cycli, 7(4,3) T-cycli. De bron is register indirect geadresseerd.

LD A,I (load)

Laadt de accumulator met de inhoud van het interrupt-vector-register I.

1 1 1 0 1 1 0 1 ED
0 1 0 1 0 1 1 1 57

carry flag : onbeïnvloed
zero flag : geset als inhoud I nul is, anders gereset
parity/overflow : kopie van IFF2
sign flag : geset als inhoud I negatief is, anders gereset
subtract flag : gereset
half carry flag : gereset

2 M-cycli, 9(4,5) T-cycli. De bron is register geadresseerd.

LD A,(nn) (load)

Laadt de accumulator met de inhoud van de door nn aangewezen geheugenlocatie.

0 0 1 1 1 0 1 0 3A
< - - -n- - - > lagere orde-deel adres
< - - -n- - - > hogere orde-deel adres

Vlaggen onbeïnvloed.

4 M-cycli, 13(4,3,3,3) T-cycli. De bron is uitgebreid geadresseerd.

LD A,R (load)

Laadt de accumulator met de inhoud van refresh register R.

```
1 1 1 0 1 1 0 1   ED
0 1 0 1 1 1 1 1   5F
```

carry flag : onbeïnvloed
zero flag : geset als inhoud R nul is, anders gereset
parity/overflow : kopie van IFF2
sign flag : geset als inhoud R negatief is, anders gereset
subtract flag : gereset
half carry flag : gereset

2 M-cycli, 9(4,5) T-cycli. De bron is register geadresseerd.

LD (BC),A (load)

Laadt de inhoud van de accumulator in de door registerpaar BC aangewezen geheugenlocatie.

```
0 0 0 0 0 0 1 0   02
```

Vlaggen niet beïnvloed.

2 M-cycli, 7(4,3) T-cycli. De bron is register geadresseerd.

LD (DE),A (load)

Laadt de inhoud van de accumulator in de door registerpaar DE aangewezen geheugenlocatie.

```
0 0 0 1 0 0 1 0   12
```

Vlaggen niet beïnvloed.

2 M-cycli, 7(4,3) T-cycli. De bron is register geadresseerd.

LD HL,(nn) (load)

Laadt de inhoud van de door nn aangewezen geheugenlocatie in register L en die van de door nn+1 aangewezen geheugenlocatie in register H.

```
0 0 1 0 1 0 1 0   2A
< - - -n- - - >   lagere orde-deel adres
< - - -n- - - >   hogere orde-deel adres
```

Vlaggen niet beïnvloed.

5 M-cycli, 16(4,3,3,3,3) T-cycli. De bron is uitgebreid geadresseerd.

LD (HL),n (load)

Laadt data n in de geheugenlocatie waarnaar de inhoud van registerpaar HL wijst.

```
0 0 1 1 0 1 1 0   36
< - - -n- - - >   data
```

Vlaggen niet beïnvloed.

3 M-cycli, 10(4,3,3) T-cycli. De bron is onmiddellijk geadresseerd.

LD (HL),r (load)

Laadt de inhoud van het door r te specificeren register in de geheugenlocatie waarnaar de inhoud van registerpaar HL wijst.

r kan zijn: A, B, C, D, E, H of L

0 1 1 1 0 < r >	r=A	1 1 1	77
	r=B	0 0 0	70
	r=C	0 0 1	71
	r=D	0 1 0	72
	r=E	0 1 1	73
	r=H	1 0 0	74
	r=L	1 0 1	75

Vlaggen niet beïnvloed.

2 M-cycli, 7(4,3) T-cycli. De bron is register geadresseerd.

LD I,A (load)

Laadt de inhoud van de accumulator in het interrupt-vector-register I.

1 1 1 0 1 1 0 1	ED
0 1 0 0 0 1 1 1	47

Vlaggen niet beïnvloed.

2 M-cycli, 9(4,5) T-cycli. De bron is register geadresseerd.

LD IX,nn (load)

Laadt het IX-register met de door nn te specificeren data.

1 1 0 1 1 1 0 1	DD
0 0 1 0 0 0 0 1	21
< - - -n- - - >	lagere orde-deel data
< - - -n- - - >	hogere orde-deel data

Vlaggen niet beïnvloed.

4 M-cycli, 14(4,4,3,3) T-cycli. De data is onmiddellijk uitgebreid geadresseerd.

LD IX,(nn) (load)

Laadt register X met de inhoud van de door nn aangewezen geheugenlocatie en register I met die van de door nn+1 aangewezen geheugenlocatie.

1 1 0 1 1 1 0 1	DD
0 0 1 0 1 0 1 0	2A
< - - -n- - - >	lagere orde-deel adres
< - - -n- - - >	hogere orde-deel adres

Vlaggen niet beïnvloed.

6 M-cycli, 20(4,4,3,3,3,3) T-cycli. De bron is uitgebreid geadresseerd.

LD (IX+d),n (load)

Laadt de door n te specificeren data in de geheugenlocatie waarnaar de inhoud van IX+d wijst.

```
1 1 0 1 1 1 0 1   DD
0 0 1 1 0 1 1 0   36
< - - -d- - - >   verplaatsing (twee-complement)
< - - -n- - - >   data
```

Vlaggen niet beïnvloed.

5 M-cycli, 19(4,4,3,5,3) T-cycli. De data is onmiddellijk geadresseerd.

LD (IX+d),r (load)

Laadt de inhoud van het door r te specificeren register in de geheugenlocatie waarnaar de inhoud van IX+d wijst.

r kan zijn: A, B, C, D, E, H of L

```
1 1 0 1 1 1 0 1   DD
0 1 1 1 0 < r >   r=A  1 1 1   77
                  r=B  0 0 0   70
                  r=C  0 0 1   71
                  r=D  0 1 0   72
                  r=E  0 1 1   73
                  r=H  1 0 0   74
                  r=L  1 0 1   75
< - - -d- - - >   verplaatsing (twee-complement)
```

Vlaggen niet beïnvloed.

5 M-cycli, 19(4,4,3,5,3) T-cycli. De bron is register geadresseerd.

LD IY,nn (load)

Laadt het IY-register met de door nn te specificeren data.

```
1 1 1 1 1 1 0 1   FD
0 0 1 0 0 0 0 1   21
< - - -n- - - >   lagere orde-deel data
< - - -n- - - >   hogere orde-deel data
```

Vlaggen niet beïnvloed.

4 M-cycli, 14(4,4,3,3) T-cycli. De data is onmiddellijk uitgebreid geadresseerd.

LD IY,(nn) (load)

Laadt register Y met de inhoud van de door nn aangewezen geheugenlocatie en register I met die van de door nn+1 aangewezen geheugenlocatie.

```
1 1 1 1 1 1 0 1   FD
0 0 1 0 1 0 1 0   2A
< - - -n- - - >   lagere orde-deel adres
< - - -n- - - >   hogere orde-deel adres
```

Vlaggen niet beïnvloed.

6 M-cycli, 20(4,4,3,3,3,3) T-cycli. De bron is uitgebreid geadresseerd.

LD (IY+d),n (load)

Laadt de door n te specificeren data in de geheugenlocatie waarnaar de inhoud van (IY+d) wijst.

```
1 1 1 1 1 1 0 1   FD
0 0 1 1 0 1 1 0   36
< - - -d- - - >   verplaatsing (twee-complement)
< - - -n- - - >   data
```

Vlaggen niet beïnvloed.

5 M-cycli, 19(4,4,3,5,3) T-cycli. De data is onmiddellijk geadresseerd.

LD (IY+d),r (load)

Laadt de inhoud van het door r te specificeren register in de geheugenlocatie waarnaar de inhoud van IY+d wijst.

r kan zijn: A, B, C, D, E, H of L

```
1 1 1 1 1 1 0 1   FD
0 1 1 1 0 < r >   r=A  1 1 1   77
                  r=B  0 0 0   70
                  r=C  0 0 1   71
                  r=D  0 1 0   72
                  r=E  0 1 1   73
                  r=H  1 0 0   74
                  r=L  1 0 1   75
< - - -d- - - >   verplaatsing (twee-complement)
```

Vlaggen niet beïnvloed.

5 M-cycli, 19(4,4,3,5,3) T-cycli. De bron is register geadresseerd.

LD (nn),A (load)

Laadt de inhoud van de accumulator in de door nn te specificeren geheugenlocatie.

```
0 0 1 1 0 0 1 0   32
< - - -n- - - >   lagere orde-deel adres
< - - -n- - - >   hogere orde-deel adres
```

Vlaggen niet beïnvloed.

4 M-cycli, 13(4,3,3,3) T-cycli. De bron is register geadresseerd.

LD (nn),HL (load)

Laadt de inhoud van register L naar de door nn te specificeren geheugenlocatie en die van H naar de geheugenlocatie waarnaar nn+1 wijst.

```
0 0 1 0 0 0 1 0   22
< - - -n- - - >   lagere orde-deel adres
< - - -n- - - >   hogere orde-deel adres
```

Vlaggen niet beïnvloed.

5 M-cycli, 16(4,3,3,3,3) T-cycli. De bron is register geadresseerd.

LD (nn),IX (load)

Laadt de inhoud van register X naar de door nn te specificeren geheugenlocatie en die van I naar de geheugenlocatie waarnaar nn+1 wijst.

```
1 1 0 1 1 1 0 1   DD
0 0 1 0 0 0 1 0   22
< - - -n- - - >   lagere orde-deel adres
< - - -n- - - >   hogere orde-deel adres
```

Vlaggen niet beïnvloed.

6 M-cycli, 20(4,4,3,3,3,3) T-cycli. De bron is register geadresseerd.

LD (nn),IY (load)

Laadt de inhoud van register Y naar de door nn te specificeren geheugenlocatie en die van I naar de geheugenlocatie waarnaar nn+1 wijst.

```
1 1 1 1 1 1 0 1   FD
0 0 1 0 0 0 1 0   22
< - - -n- - - >   lagere orde-deel adres
< - - -n- - - >   hogere orde-deel adres
```

Vlaggen niet beïnvloed.

6 M-cycli, 20(4,4,3,3,3,3) T-cycli. De bron is register geadresseerd.

LD (nn),ss (load)

Laadt het lagere orde-deel van de inhoud van het door ss te specificeren registerpaar naar de door nn te specificeren geheugenlocatie en die van het hogere orde-deel naar de geheugenlocatie waarnaar nn+1 wijst.

ss kan zijn: BC, DE, HL of SP

```
1 1 1 0 1 1 0 1   ED
0 1 s s 0 0 1 1   ss=BC  0 0   43
                  ss=DE  0 1   53
                  ss=HL  1 0   63
                  ss=SP  1 1   73
< - - -n- - - >   lagere orde-deel adres
< - - -n- - - >   hogere orde-deel adres
```

6 M-cycli, 20(4,4,3,3,3,3) T-cycli. De bron is register geadresseerd.

LD R,A (load)

Laadt de inhoud van de accumulator in het refresh register R.

```
1 1 1 0 1 1 0 1   ED
0 1 0 0 1 1 1 1   4F
```

Vlaggen niet beïnvloed.

2 M-cycli, 9(4,5) T-cycli. De bron is register geadresseerd.

LD r,(HL) (load)

Laadt het door r te specificeren register met de inhoud van de geheugenlocatie waarnaar HL wijst.

r kan zijn: A, B, C, D, E, H of L

```
0 1 < r > 1 1 0   r=A  1 1 1   7E
                  r=B  0 0 0   46
                  r=C  0 0 1   4E
                  r=D  0 1 0   56
                  r=E  0 1 1   5E
                  r=H  1 0 0   66
                  r=L  1 0 1   6E
```

Vlaggen niet beïnvloed.

2 M-cycli, 7(4,3) T-cycli. De bron is register indirect geadresseerd.

LD r,(IX+d) (load)

Laadt het door r te specificeren register met de inhoud van de geheugenlocatie waarnaar IX+d wijst.

r kan zijn: A, B, C, D, E, H of L

```
1 1 0 1 1 1 0 1   DD
0 1 < r > 1 1 0   r=A  1 1 1   7E
                  r=B  0 0 0   46
                  r=C  0 0 1   4E
                  r=D  0 1 0   56
                  r=E  0 1 1   5E
                  r=H  1 0 0   66
                  r=L  1 0 1   6E
< - - -d- - - >   verplaatsing (twee-complement)
```

Vlaggen niet beïnvloed.

5 M-cycli, 19(4,4,3,5,3) T-cycli. De bron is geïndexeerd geadresseerd.

LD r,(IY+d) (load)

Laadt het door r te specificeren register met de inhoud van de geheugenlocatie waarnaar IY+d wijst.

r kan zijn: A, B, C, D, E, H of L

```
1 1 1 1 1 1 0 1   FD
0 1 < r > 1 1 0   r=A  1 1 1   7E
                  r=B  0 0 0   46
                  r=C  0 0 1   4E
                  r=D  0 1 0   56
                  r=E  0 1 1   5E
                  r=H  1 0 0   66
                  r=L  1 0 1   6E
< - - -d- - - >   verplaatsing (twee-complement)
```

Vlaggen niet beïnvloed.

5 M-cycli, 19(4,4,3,5,3) T-cycli. De bron is geïndexeerd geadresseerd.

LD r,n (load)

Laadt het door r te specificeren register met data n.

r kan zijn: A, B, C, D, E, H of L

```
0 0 < r > 1 1 0   r=A  1 1 1   3E
                  r=B  0 0 0   06
                  r=C  0 0 1   0E
                  r=D  0 1 0   16
                  r=E  0 1 1   1E
                  r=H  1 0 0   26
                  r=L  1 0 1   2E
< - - -n- - - >   data
```

Vlaggen niet beïnvloed.

2 M-cycli, 7(4,3) T-cycli. De data is onmiddellijk geadresseerd.

LD r,r' (load)

Laadt het door r te specificeren register met de inhoud van het door r' te specificeren register.

r en r' kunnen zijn: A, B, C, D, E, H of L

```
0 1 < r > < r'>     r of r'=A    1 1 1
                    r of r'=B    0 0 0
                    r of r'=C    0 0 1
                    r of r'=D    0 1 0
                    r of r'=E    0 1 1
                    r of r'=H    1 0 0
                    r of r'=L    1 0 1
```

r= \ r'	A	B	C	D	E	H	L
A	7F	78	79	7A	7B	7C	7D
B	47	40	41	42	43	44	45
C	4F	48	49	4A	4B	4C	4D
D	57	50	51	52	53	54	55
E	5F	58	59	5A	5B	5C	5D
H	67	60	61	62	63	64	65
L	6F	68	69	6A	6B	6C	6D

Vlaggen niet beïnvloed.
1 M-cyclus, 4 T-cycli. Bron en bestemming zijn register geadresseerd.

LD SP,HL (load)

Laadt registerpaar SP met de inhoud van registerpaar HL.

```
1 1 1 1 1 0 0 1  F9
```

Vlaggen niet beïnvloed.
1 M-cyclus, 6 T-cycli. De bron is register geadresseerd.

LD SP,IX (load)

Laadt registerpaar SP met de inhoud van registerpaar IX.

```
1 1 0 1 1 1 0 1  DD
1 1 1 1 1 0 0 1  F9
```

Vlaggen niet beïnvloed.

2 M-cycli, 10(4,6) T-cycli. De bron is register geadresseerd.

LD SP,IY (load)

Laadt registerpaar SP met de inhoud van registerpaar IY.

```
1 1 1 1 1 1 0 1  FD
1 1 1 1 1 0 0 1  F9
```

Vlaggen niet beïnvloed.

2 M-cycli, 10(4,6) T-cycli. De bron is register geadresseerd.

LD ss,nn (load)

Laadt het door ss te specificeren registerpaar met data nn.

ss kan zijn: BC, DE, HL of SP

```
0 0 s s 0 0 0 1     ss=BC   0 0   01
                    ss=DE   0 1   11
                    ss=HL   1 0   21
                    ss=SP   1 1   31
< - - -n- - - >     lagere orde-deel data
< - - -n- - - >     hogere orde-deel data
```

Vlaggen niet beïnvloed.

3 M-cycli, 10(4,3,3) T-cycli. De data is onmiddellijk uitgebreid geadresseerd.

LD ss,(nn) (load)

Laadt het lagere orde-deel van het door ss te specificeren registerpaar met de inhoud van de geheugenlocatie nn en het hogere orde-deel met dat van nn+1.

ss kan zijn: BC, DE, HL of SP

```
1 1 1 0 1 1 0 1   ED
0 1 s s 1 0 1 1   ss=BC  0 0   4B
                  ss=DE  0 1   5B
                  ss=HL  1 0   6B
                  ss=SP  1 1   7B
< - - -n- - - >   lagere orde-deel adres
< - - -n- - - >   hogere orde-deel adres
```

Vlaggen niet beïnvloed.

6 M-cycli, 20(4,4,3,3,3,3) T-cycli. De bron is uitgebreid geadresseerd.

LDD (load decrement)

Laadt de inhoud van de geheugenlocatie waarnaar registerpaar HL wijst in de geheugenlocatie waarnaar registerpaar DE wijst. Daarna worden de inhouden van HL, DE en het als teller te gebruiken registerpaar BC met één verlaagd.

```
1 1 1 0 1 1 0 1   ED
1 0 1 0 1 0 0 0   A8
```

carry flag : onbeïnvloed
zero flag : onbeïnvloed
parity/overflow : geset als BC ongelijk aan nul is, anders gereset
sign flag : onbeïnvloed
subtract flag : gereset
half carry flag : gereset

4 M-cycli, 16(4,4,3,5) T-cycli.

LDDR (load decrement repeat)

Laadt de inhoud van de geheugenlocatie waarnaar registerpaar HL wijst in de geheugenlocatie waarnaar registerpaar DE wijst. Daarna worden de inhouden van HL, DE en de teller BC met één verlaagd. Als de inhoud van BC hierna niet nul is, herhaalt de Z80 de instructie door de inhoud van de PC met twee te verminderen.

```
1 1 1 0 1 1 0 1   ED
1 0 1 1 1 0 0 0   B8
```

carry flag : onbeïnvloed
zero flag : onbeïnvloed
parity/overflow : gereset
sign flag : onbeïnvloed
subtract flag : gereset
half carry flag : gereset
Als BC nul is: 4 M-cycli, 16(4,4,3,5) T-cycli
Als BC ongelijk aan nul is: 5 M-cycli, 21(4,4,3,5,5) T-cycli

LDI (load increment)

Laadt de inhoud van de geheugenlocatie waarnaar registerpaar HL wijst in de geheugenlocatie waarnaar registerpaar DE wijst. Daarna worden de inhouden van HL en DE met één verhoogd en die van het als teller te gebruiken registerpaar BC met één verlaagd.

```
1 1 1 0 1 1 0 1   ED
1 0 1 0 0 0 0 0   A0
```

carry flag : onbeïnvloed
zero flag : onbeïnvloed
parity/overflow : geset als BC ongelijk aan nul is, anders gereset
sign flag : onbeïnvloed
subtract flag : gereset
half carry flag : gereset

4 M-cycli, 16(4,4,3,5) T-cycli.

LDIR (load increment repeat)

Laadt de inhoud van de geheugenlocatie waarnaar registerpaar HL wijst in de geheugenlocatie waarnaar registerpaar DE wijst. Daarna worden de inhouden van HL en DE met één verhoogd en die van teller BC met één verlaagd. Als de inhoud van BC hierna niet nul is, herhaalt de Z80 de instructie door de inhoud van de PC met twee te verminderen.

```
1 1 1 0 1 1 0 1   ED
1 0 1 1 0 0 0 0   B0
```

carry flag : onbeïnvloed
zero flag : onbeïnvloed
parity/overflow : gereset
sign flag : onbeïnvloed
subtract flag : gereset
half carry flag : gereset

Als BC nul is: 4 M-cycli, 16(4,4,3,5) T-cycli
Als BC ongelijk aan nul is: 5 M-cycli, 21(4,4,3,5,5) T-cycli

NEG (negate)

Trekt de inhoud van de accumulator af van nul en zet het resultaat, de twee-complements voorstelling van de accumulator-inhoud, in de accumulator. (Van de getallen 00H en 80H is het resultaat van de bewerking weer 00H en 80H.)

```
1 1 1 0 1 1 0 1   ED
0 1 0 0 0 1 0 0   44
```

carry flag : gereset als accumulatorinhoud 00H is, anders geset
zero flag : geset als het resultaat nul is, anders gereset
parity/overflow : geset als de accumulatorinhoud 80H is, anders gereset
sign flag : geset als het resultaat negatief is, anders gereset

NEG (vervolg)

subtract flag : geset
half carry flag : geset door een borrow van bit 4, anders gereset

2 M-cycli, 8(4,4) T-cycli. De accumulator is impliciet geadresseerd.

NOP (no operation)

De Z80 doet 1 machinecyclus lang niets.

```
0 0 0 0 0 0 0 0   00
```

Vlaggen niet beïnvloed.

1 M-cyclus, 4 T-cycli.

OR s

Logische OF van de inhoud van de accumulator en de door s te specificeren operand. Het resultaat komt in de accumulator.
s kan zijn: r, n, (HL), (IX+d) of (IY+d)

r	1 0 1 1 0 < r >	r=A 1 1 1	B7
		r=B 0 0 0	B0
		r=C 0 0 1	B1
		r=D 0 1 0	B2
		r=E 0 1 1	B3
		r=H 1 0 0	B4
		r=L 1 0 1	B5
n	1 1 1 1 0 1 1 0		F6
	< - - -n- - - >		data
(HL)	1 0 1 1 0 1 1 0		B6
(IX+d)	1 1 0 1 1 1 0 1		DD
	1 0 1 1 0 1 1 0		B6
	< - - -d- - - >		verplaatsing (twee-complement)
(IY+d)	1 1 1 1 1 1 0 1		FD
	1 0 1 1 0 1 1 0		B6
	< - - -d- - - >		verplaatsing (twee-complement)

carry flag : gereset
zero flag : geset als het resultaat nul is, anders gereset
parity/overflow : geset bij even pariteit, anders gereset
sign flag : geset als het resultaat negatief is, anders gereset
subtract flag : gereset
half carry flag : gereset

	M-cycli	T-cycli	adresseringswijze
r	1	4	register
n	2	7(4,3)	onmiddellijk
(HL)	2	7(4,3)	register indirect
(IX+d)	5	19(4,4,3,5,3)	geïndexeerd
(IY+d)	5	19(4,4,3,5,3)	geïndexeerd

OTDR (output decrement repeat)

Slaat de inhoud van de geheugenlocatie waarnaar registerpaar HL wijst op in de Z80. Vervolgens wordt de inhoud van het als teller te gebruiken register B met één verlaagd. De inhoud van register C, het poortadres, komt op het lagere orde-deel van de adresbus (A0 t/m A7) en op het hogere deel (A8 t/m A15) komt de inhoud van B. De Z80 schrijft het eerder opgeslagen byte naar de door C geselecteerde poort. Daarna wordt de inhoud van registerpaar HL met één verlaagd. Is de inhoud van teller B nu niet nul dan herhaalt de Z80 de instructie door de inhoud van de PC met twee te verminderen.

1 1 1 0 1 1 0 1	ED
1 0 1 1 1 0 1 1	BB

carry flag : onbeïnvloed
zero flag : geset
parity/overflow : ongedefinieerd
sign flag : ongedefinieerd
subtract flag : geset
half carry flag : ongedefinieerd

Als B nul is: 4 M-cycli, 16(4,5,3,4) T-cycli
Als B ongelijk aan nul is: 5 M-cycli, 21(4,5,3,4,5) T-cycli

OTIR (output increment repeat)

Slaat de inhoud van de geheugenlocatie waarnaar registerpaar HL wijst op in de Z80. Vervolgens wordt de inhoud van het als teller te gebruiken register B met één verlaagd. De inhoud van register C, het poortadres, komt op het lagere orde-deel van de adresbus (A0 t/m A7) en op het hogere orde-deel (A8 t/m A15) komt de inhoud van B. De Z80 schrijft het eerder opgeslagen byte naar de door C geselecteerde poort. Daarna wordt de inhoud van registerpaar HL met één verhoogd. Is de inhoud van teller B nu ongelijk aan nul dan herhaalt de Z80 de instructie door de inhoud van de PC met twee te verminderen.

```
1 1 1 0 1 1 0 1   ED
1 0 1 1 0 0 1 1   B3
```

carry flag : onbeïnvloed
zero flag : geset
parity/overflow : ongedefinieerd
sign flag : ongedefinieerd
subtract flag : geset
half carry flag : ongedefinieerd

Als B nul is: 4 M-cycli, 16(4,5,3,4) T-cycli
Als B ongelijk aan nul is: 5 M-cycli, 21(4,5,3,4,5) T-cycli

OUT (C),r (output)

Zet de inhoud van het door r te specificeren register op de door C aangewezen output-poort. De Z80 plaatst daartoe de inhoud van C op het lagere orde-deel van de adresbus (A0 t/m A7), die van register B op het hogere orde-deel (A8 t/m A15) en de inhoud van het door r te specificeren register op de databus.

r kan zijn: A, B, C, D, E, H of L

```
1 1 1 0 1 1 0 1   ED
0 1 < r > 0 0 1
```

r=A	1 1 1	79
r=B	0 0 0	41
r=C	0 0 1	49
r=D	0 1 0	51
r=E	0 1 1	59
r=H	1 0 0	61
r=L	1 0 1	69

Vlaggen niet beïnvloed.

3 M-cycli, 12(4,4,4) T-cycli. De poort is register indirect geadresseerd.

OUT (n),A (output)

Zet de inhoud van de accumulator op de door n te specificeren output-poort. De Z80 plaatst daartoe n op het lagere orde-deel van de adresbus (A0 t/m A8). De inhoud van de accumulator komt op het hogere orde-deel van de adresbus (A8 t/m A15) en op de databus.

```
1 1 0 1 0 0 1 1   D3
< - - -n- - - >   poort
```

Vlaggen niet beïnvloed.

3 M-cycli, 11(4,3,4) T-cycli. De poort is onmiddellijk geadresseerd.

OUTD (output decrement)

Slaat de inhoud van de geheugenlocatie waarnaar registerpaar HL wijst op in de Z80. Vervolgens wordt de inhoud van het als teller te gebruiken register B met één verlaagd. De inhoud van register C, het poortadres, komt op het lagere orde-deel van de adresbus (A0 t/m A7) en op het hogere orde-deel (A8 t/m A15) komt de inhoud van B. De Z80 schrijft het eerder opgeslagen byte naar de door C geselecteerde poort. Daarbij wordt de inhoud van registerpaar HL met één verlaagd.

```
1 1 1 0 1 1 0 1   ED
1 0 1 0 1 0 1 1   AB
```

carry flag	:	onbeïnvloed
zero flag	:	geset als B nul is, anders gereset
parity/overflow	:	ongedefinieerd
sign flag	:	ongedefinieerd
subtract flag	:	geset
half carry flag	:	ongedefinieerd

4 M-cycli, 16(4,5,3,4) T-cycli.

OUTI (output increment)

Slaat de inhoud van de geheugenlocatie waarnaar registerpaar HL wijst op in de Z80. Vervolgens wordt de inhoud van het als teller te gebruiken register B met één verlaagd. De inhoud van register C, het poortadres, komt op het lagere orde-deel van de adresbus (A0 t/m A7) en op het hogere orde-deel (A8 t/m A15) komt de inhoud van B. De Z80 schrijft het eerder opgeslagen byte naar de door C geselecteerde poort. Daarna wordt de inhoud van registerpaar HL met één verhoogd.

```
1 1 1 0 1 1 0 1   ED
1 0 1 0 0 0 1 1   A3
```

carry flag	:	onbeïnvloed
zero flag	:	geset als B nul is, anders gereset
parity/overflow	:	ongedefinieerd
sign flag	:	ongedefinieerd
subtract flag	:	geset
half carry flag	:	ongedefinieerd

4 M-cycli, 16(4,5,3,4) T-cycli.

POP IX

Plaatst de top van de stapel in registerpaar IX. De Z80 zet de inhoud van de geheugenlocatie waarnaar SP wijst in register X. De inhoud van SP wordt met één verhoogd en de inhoud van de geheugenlocatie waarnaar SP nu wijst wordt in register I gezet. Vervolgens wordt SP nogmaals met één verhoogd en wijst, net als aan het begin van de instructie, naar de laatste bezette plaats van de stapel.

```
1 1 0 1 1 1 0 1   DD
1 1 1 0 0 0 0 1   E1
```

Vlaggen niet beïnvloed.

4 M-cycli, 14(4,4,3,3) T-cycli. De bron is impliciet geadresseerd.

POP IY

Plaatst de top van de stapel in registerpaar IY. De Z80 zet de inhoud van de geheugenlocatie waarnaar SP wijst in register Y. De inhoud van SP wordt met één verhoogd en de geheugenlocatie waarnaar SP nu wijst wordt in register I gezet. Vervolgens wordt SP nogmaals met één verhoogd en wijst, net als aan het begin van de instructie, naar de laatste bezette plaats van de stapel.

```
1 1 1 1 1 1 0 1   FD
1 1 1 0 0 0 0 1   E1
```

Vlaggen niet beïnvloed.

4 M-cycli, 14(4,4,3,3) T-cycli. De bron is impliciet geadresseerd.

POP qq

Plaatst de top van de stapel in het door qq te specificeren registerpaar. De Z80 zet de inhoud van de geheugenlocatie waarnaar SP wijst in het lagere orde-deel van qq. De inhoud van SP wordt met één verhoogd en de inhoud van de geheugenlocatie waarnaar SP nu wijst wordt in het hogere orde-deel van qq gezet. Vervolgens wordt SP nogmaals met één verhoogd en wijst, net als aan het begin van de instructie, naar de laatst bezette plaats van de stapel.

qq kan zijn: AF, BC, DE of HL

```
1 1 q q 0 0 0 1   qq=AF  1 1   F1
                  qq=BC  0 0   C1
                  qq=DE  0 1   D1
                  qq=HL  1 0   E1
```

Vlaggen niet beïnvloed.

3 M-cycli, 10(4,3,3) T-cycli. De bron is impliciet geadresseerd.

PUSH IX

Zet de inhoud van registerpaar IX op de stapel. De Z80 verlaagt de inhoud van SP en zet de inhoud van register I in de geheugenlocatie waarnaar SP wijst. Vervolgens wordt SP nogmaals met één verlaagd en de inhoud van register X komt in de geheugenlocatie waarnaar SP nu wijst. Aan het begin en het einde van de instructie wijst SP naar de laatst bezette plaats van de stapel.

```
1 1 0 1 1 1 0 1   DD
1 1 1 0 0 1 0 1   E5
```

Vlaggen niet beïnvloed.

4 M-cycli, 15(4,5,3,3) T-cycli. De bron is register geadresseerd.

PUSH IY

Zet de inhoud van registerpaar IY op de stapel. De Z80 verlaagt de inhoud van SP en zet de inhoud van register I in de geheugenlocatie waarnaar SP wijst. Vervolgens wordt SP nogmaals met één verlaagd en de inhoud van register Y komt in de geheugenlocatie waarnaar SP nu wijst. Aan het begin en het einde van de instructie wijst SP naar de laatst bezette plaats van de stapel.

```
1 1 1 1 1 1 0 1   FD
1 1 1 0 0 1 0 1   E5
```

Vlaggen niet beïnvloed.

4 M-cycli, 15(4,5,3,3) T-cycli. De bron is register geadresseerd.

PUSH qq

Zet de inhoud van het door qq te specificeren registerpaar op de stapel. De Z80 verlaagt de inhoud van SP en zet de inhoud van het hogere orde-deel van qq in de geheugenlocatie waarnaar SP wijst. Vervolgens wordt SP nogmaals met één verlaagd en de inhoud van het lagere orde-deel van qq komt in de geheugenlocatie waarnaar SP nu wijst. Aan het begin en het einde van de instructie wijst SP naar de laatst bezette plaats van de stapel.

qq kan zijn: AF, BC, DE of HL

1 1 q q 0 1 0 1	qq=AF	1 1	F5
	qq=BC	0 0	C5
	qq=DE	0 1	D5
	qq=HL	1 0	E5

Vlaggen niet beïnvloed.

3 M-cycli, 11(5,3,3) T-cycli. De bron is register geadresseerd.

RES b,m (reset)

Reset het door b te specificeren bit van de door m te specificeren operand.

m kan zijn: r, (HL), (IX+d) of (IY+d)
bit 0 is het laagste bit

r	1 1 0 0 1 0 1 1	CB		
	1 0 < b > < r >	r=A	1 1 1	
		r=B	0 0 0	
		r=C	0 0 1	
		r=D	0 1 0	
		r=E	0 1 1	
		r=H	1 0 0	
		r=L	1 0 1	
(HL)	1 1 0 0 1 0 1 1	CB	bit 0=	0 0 0
	1 0 < b > 1 1 0		bit 1=	0 0 1

RES b,m (vervolg)

			bit 2=	0 1 0
			bit 3=	0 1 1
			bit 4=	1 0 0
			bit 5=	1 0 1
			bit 6=	1 1 0
(IX+d)	1 1 0 1 1 1 0 1	DD	bit 7=	1 1 1
	1 1 0 0 1 0 1 1	CB		
	< - - -d- - - >	verplaatsing (twee-complement)		
	1 0 < b > 1 1 0			
(IY+d)	1 1 1 1 1 1 0 1	FD		
	1 1 0 0 1 0 1 1	CB		
	< - - -d- - - >	verplaatsing (twee-complement)		
	1 0 < b > 1 1 0			

bit	A	B	C	D	E	H	L	(IX+d) en (IY+d) en (HL)
0	87	80	81	82	83	84	85	86
1	8F	88	89	8A	8B	8C	8D	8E
2	97	90	91	92	93	94	95	96
3	9F	98	99	9A	9B	9C	9D	9E
4	A7	A0	A1	A2	A3	A4	A5	A6
5	AF	A8	A9	AA	AB	AC	AD	AE
6	B7	B0	B1	B2	B3	B4	B5	B6
7	BF	B8	B9	BA	BB	BC	BD	BE

	M-cycli	T-cycli	adresseringswijze
r	2	8(4,4)	register
(HL)	4	15(4,4,4,3)	register indirect
(IX+d)	6	23(4,4,3,5,4,3)	geïndexeerd
(IY+d)	6	23(4,4,3,5,4,3)	geïndexeerd

RET (return)

Terugkeer van een subroutine. Op de manier zoals bij de POP-instructie is beschreven wordt de top van de stapel in de PC gezet. (Door de CALL waarmee de subroutine werd aangeroepen werd de PC op de top van de stapel gezet.)

1 1 0 0 1 0 0 1 C9

Vlaggen niet beïnvloed.

3 M-cycli, 10(4,3,3) T-cycli.

RET cc (return)

Terugkeer van een subroutine als de door cc te specificeren conditie waar is. Op de manier zoals bij de POP-instructie is beschreven wordt de top van de stapel in de PC gezet. (Door de CALL waarmee de subroutine werd aangeroepen werd de PC op de top van de stapel gezet.) Als conditie cc niet waar is gaat de Z80 verder met de volgende instructie.

cc kan zijn: NZ, Z, NC, C, PO, PE, P of M

1 1 <c c> 0 0 0

cc=NZ	non zero	0 0 0	C0	(zero flag)
cc=Z	zero	0 0 1	C8	
cc=NC	non carry	0 1 0	D0	(carry)
cc=C	carry	0 1 1	D8	
cc=PO	pariteit oneven	1 0 0	E0	(P/V-vlag)
cc=PE	pariteit even	1 0 1	E8	
cc=P	plus	1 1 0	F0	(sign flag)
cc=M	min	1 1 1	F8	

Vlaggen niet beïnvloed.

Als cc waar is: 3 M-cycli, 11(5,3,3) T-cycli
Als cc niet waar is: 1 M-cyclus, 5 T-cycli

RETI (return from interrupt)

Terugkeer van een maskeerbare interrupt. Op de manier zoals bij de POP-instructie is beschreven wordt de top van de stapel in de PC gezet. De Zilog-randapparatuur-chips zien aan het verschijnen van ED en 4D op de databus tijdens M1 cycli dat de interrupt service subroutine ten einde is. Als maskeerbare interrupts tijdens de interrupt service niet waren toegestaan en na de terugkeer wel toegestaan zijn, moet RETI worden voorafgegaan door EI (enable interrupt). Bij gebruik van een daisy chain regelt de instructie de afwerking van geneste interrupts met verschillende prioriteiten door de IEO van het apparaat waarvan de service routine ten einde is te setten.

1 1 1 0 1 1 0 1 ED
0 1 0 0 1 1 0 1 4D

Vlaggen niet beïnvloed.

4 M-cycli, 14(4,4,3,3) T-cycli.

RETN (return from non maskable interrupt)

Terugkeer van een niet maskeerbare interrupt. Op de manier zoals bij de POP-instructie is beschreven wordt de top van de stapel in de PC gezet. Interrupt-flipflop IFF2 wordt gekopieerd in IFF1 zodat wat het al dan niet toestaan van maskeerbare interrupts betreft de toestand gelijk is aan die van voor de niet maskeerbare interrupt.

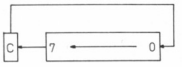
1 1 1 0 1 1 0 1 ED
0 1 0 0 0 1 0 1 45

Vlaggen niet beïnvloed.

4 M-cycli, 14(4,4,3,3) T-cycli.

RL m (rotate left)

Roteert de door m te specificeren operand 1 bit links door de carry. De inhoud van bit 7 komt in de carry en de inhoud van de carry in bit 0. Bit 0 is het laagste bit.



m kan zijn: r, (HL), (IX+d) of (IY+d)

r	1 1 0 0 1 0 1 1	CB			
	0 0 0 1 0 < r >	r=A	1 1 1	17
		r=B	0 0 0	10
		r=C	0 0 1	11
		r=D	0 1 0	12
		r=E	0 1 1	13
		r=H	1 0 0	14
		r=L	1 0 1	15
(HL)	1 1 0 0 1 0 1 1	CB		
	0 0 0 1 0 1 1 0	16		
(IX+d)	1 1 0 1 1 1 0 1	DD		
	1 1 0 0 1 0 1 1	CB		
	< - - -d- - - >	verplaatsing (twee-complement)		
	0 0 0 1 0 1 1 0	16		
(IY+d)	1 1 1 1 1 1 0 1	FD		
	1 1 0 0 1 0 1 1	CB		
	< - - -d- - - >	verplaatsing (twee-complement)		
	0 0 0 1 0 1 1 0	16		

carry flag : bevat bit 7 van m
zero flag : geset als het resultaat nul is, anders gereset
parity/overflow : geset door even pariteit, anders gereset
sign flag : geset als het resultaat negatief is, anders gereset
subtract flag : gereset

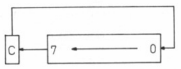
RL m (vervolg)

half carry flag : gereset

	M-cycli	T-cycli	adresseringswijze
r	2	8(4,4)	register
(HL)	4	15(4,4,4,3)	register indirect
(IX+d)	6	23(4,4,4,3,5,4,3)	geïndexeerd
(IY+d)	6	23(4,4,4,3,5,4,3)	geïndexeerd

RLA (rotate left accumulator)

Roteert de inhoud van de accumulator 1 bit links door de carry. Bit 7 van de accumulator komt in de carry en de carry komt in bit 0 van de accumulator. Bit 0 is het laagste bit.



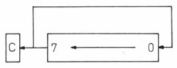
0 0 0 1 0 1 1 1 17

carry flag : bevat bit 7 van de accumulator
zero flag : niet beïnvloed
parity/overflow : niet beïnvloed
sign flag : niet beïnvloed
subtract flag : gereset
half carry flag : gereset

1 M-cyclus, 4 T-cycli. De accumulator is impliciet geadresseerd.

RLC m (rotate left circular)

Roteert de door m te specificeren operand 1 bit links. De inhoud van bit 7 komt zowel in de carry als in bit 0. Bit 0 is het laagste bit.



m kan zijn: r, (HL), (IX+d) of (IY+d)

r	1 1 0 0 1 0 1 1	CB	
	0 0 0 0 0 < r >	r=A	1 1 1 07
		r=B	0 0 0 00
		r=C	0 0 1 01
		r=D	0 1 0 02
		r=E	0 1 1 03
		r=H	1 0 0 04
		r=L	1 0 1 05
(HL)	1 1 0 0 1 0 1 1	CB	
	0 0 0 0 0 1 1 0	06	
(IX+d)	1 1 0 1 1 1 0 1	DD	
	1 1 0 0 1 0 1 1	CB	
	< - - -d- - - >	verplaatsing (twee-complement)	
	0 0 0 0 0 1 1 0	06	
(IY+d)	1 1 1 1 1 1 0 1	FD	
	1 1 0 0 1 0 1 1	CB	
	< - - -d- - - >	verplaatsing (twee-complement)	
	0 0 0 0 0 1 1 0	06	

carry flag : bevat bit 7 van m
zero flag : geset als het resultaat nul is, anders gereset
parity/overflow : geset door even pariteit, anders gereset
sign flag : geset als het resultaat negatief is, anders gereset

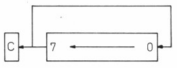
RLC m (vervolg)

subtract flag : gereset
half carry flag : gereset

	M-cycli	T-cycli	adresseringswijze
r	2	8(4,4)	register
(HL)	4	15(4,4,4,3)	register indirect
(IX+d)	6	23(4,4,3,5,4,3)	geïndexeerd
(IY+d)	6	23(4,4,3,5,4,3)	geïndexeerd

RLCA (rotate left circular accumulator)

Roteert de inhoud van de accumulator 1 bit links. De inhoud van bit 7 komt zowel in de carry als in bit 0. Bit 0 is het laagste bit.



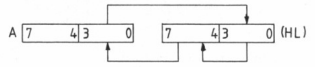
0 0 0 0 0 1 1 1 07

carry flag : bevat bit 7 van de accumulator
zero flag : niet beïnvloed
parity/overflow : niet beïnvloed
sign flag : niet beïnvloed
subtract flag : gereset
half carry flag : gereset

1 M-cyclus, 4 T-cycli. De accumulator is impliciet geadresseerd.

RLD (rotate left digit)

Zet de inhoud van de vier laagste bits van de geheugenlocatie waarnaar HL wijst in de vier hoogste bits en de voormalige inhoud van de vier hoogste bits in de laagste vier bits van de accumulator. De voormalige inhoud van de vier laagste bits van de accumulator gaan naar de vier laagste bits van de geheugenlocatie waarnaar HL wijst. Bit 0 is het laagste bit.



1 1 1 0 1 1 0 1 ED
0 1 1 0 1 1 1 1 6F

carry flag : niet beïnvloed
zero flag : geset als inhoud accumulator nul is, anders gereset
parity/overflow : geset door even pariteit accumulator, anders gereset
sign flag : geset als inhoud accumulator nul is, anders gereset
subtract flag : gereset
half carry flag : gereset

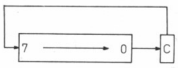
5 M-cycli, 18(4,4,3,4,3) T-cycli.

RR m (rotate right)

Roteert de door m te specificeren operand 1 bit rechts door de carry. De inhoud van bit 0 komt in de carry en de inhoud van de carry in bit 7. Bit 0 is het laagste bit.

m kan zijn: r, (HL), (IX+d) of (IY+d)

RR m (vervolg)



r 1 1 0 0 1 0 1 1 CB
0 0 0 1 1 < r > r=A 1 1 1 1F
r=B 0 0 0 18
r=C 0 0 1 19
r=D 0 1 0 1A
r=E 0 1 1 1B
r=H 1 0 0 1C
r=L 1 0 1 1D

(HL) 1 1 0 0 1 0 1 1 CB
0 0 0 1 1 1 1 0 1E

(IX+d) 1 1 0 1 1 1 0 1 DD
1 1 0 0 1 0 1 1 CB
< - - -d- - - > verplaatsing (twee-complement)
0 0 0 1 1 1 1 0 1E

(IY+d) 1 1 1 1 1 1 0 1 FD
1 1 0 0 1 0 1 1 CB
< - - -d- - - > verplaatsing (twee-complement)
0 0 0 1 1 1 1 0 1E

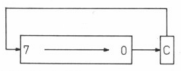
carry flag : bevat bit 0 van m
zero flag : geset als het resultaat nul is, anders gereset
parity/overflow : geset door even pariteit, anders gereset
sign flag : geset als resultaat negatief is, anders gereset
subtract flag : gereset
half carry flag : gereset

RR m (vervolg)

	M-cycli	T-cycli	adresseringswijze
r	2	8(4,4)	register
(HL)	4	15(4,4,4,3)	register indirect
(IX+d)	6	23(4,4,3,5,4,3)	geïndexeerd
(IY+d)	6	23(4,4,3,5,4,3)	geïndexeerd

RRA (rotate right accumulator)

Roteert de inhoud van de accumulator 1 bit rechts door de carry. Bit 0 van de accumulator komt in de carry en de carry komt in bit 7 van de accumulator. Bit 0 is het laagste bit.



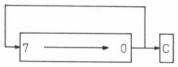
0 0 0 1 1 1 1 1 1F

carry flag : bevat bit 0 van de accumulator
zero flag : niet beïnvloed
parity/overflow : niet beïnvloed
sign flag : niet beïnvloed
subtract flag : gereset
half carry flag : gereset

1 M-cyclus, 4 T-cycli. De accumulator is impliciet geadresseerd.

RRC m (rotate right circular)

Roteert de door m te specificeren operand 1 bit rechts. De inhoud van bit 0 komt zowel in de carry als in bit 7. Bit 0 is het laagste bit.



m kan zijn: r, (HL), (IX+d) of (IY+d)

r	1 1 0 0 1 0 1 1	CB
	0 0 0 0 1 < r >	r=A 1 1 1 0F
		r=B 0 0 0 08
		r=C 0 0 1 09
		r=D 0 1 0 0A
		r=E 0 1 1 0B
		r=H 1 0 0 0C
		r=L 1 0 1 0D
(HL)	1 1 0 0 1 0 1 1	CB
	0 0 0 0 1 1 1 0	0E
(IX+d)	1 1 0 1 1 1 0 1	DD
	1 1 0 0 1 0 1 1	CB
	< - - -d- - - >	verplaatsing (twee-complement)
	0 0 0 0 1 1 1 0	0E
(IY+d)	1 1 1 1 1 1 0 1	FD
	1 1 0 0 1 0 1 1	CB
	< - - -d- - - >	verplaatsing (twee-complement)
	0 0 0 0 1 1 1 0	0E

carry flag : bevat bit 0 van m
zero flag : geset als het resultaat nul is, anders gereset
parity/overflow : geset door even pariteit, anders gereset
sign flag : geset als het resultaat negatief is, anders gereset

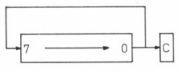
RRC m (vervolg)

subtract flag : gereset
half carry flag : gereset

	M-cycli	T-cycli	adresseringswijze
r	2	8(4,4)	register
(HL)	4	15(4,4,4,3)	register indirect
(IX+d)	6	23(4,4,3,5,4,3)	geïndexeerd
(IY+d)	6	23(4,4,3,5,4,3)	geïndexeerd

RRCA (rotate right circular accumulator)

Roteert de inhoud van de accumulator 1 bit rechts. De inhoud van bit 0 komt zowel in de carry als in bit 7. Bit 0 is het laagste bit.



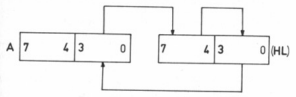
0 0 0 0 1 1 1 1 0F

carry flag : bevat bit 0 van de accumulator
zero flag : niet beïnvloed
parity/overflow : niet beïnvloed
sign flag : niet beïnvloed
subtract flag : gereset
half carry flag : gereset

1 M-cyclus, 4 T-cycli. De accumulator is impliciet geadresseerd.

RRD (rotate right digit)

Zet de inhoud van de vier hoogste bits van de geheugenlocatie waarnaar HL wijst in de vier laagste bits en de voormalige inhoud van de vier laagste bits in de vier laagste bits van de accumulator. De voormalige inhoud van de vier laagste bits van de accumulator gaat naar de vier hoogste bits van de geheugenlocatie waarnaar HL wijst. Bit 0 is het laagste bit.



1 1 1 0 1 1 0 1 ED
0 1 1 0 0 1 1 1 67

carry flag : niet beïnvloed
zero flag : geset als inhoud accumulator nul is, anders gereset
parity/overflow : geset door even pariteit accumulator, anders gereset
sign flag : geset als resultaat negatief is, anders gereset
subtract flag : gereset
half carry flag : gereset

5 M-cycli, 18(4,4,3,4,3) T-cycli.

RST p (restart)

Start opnieuw op pagina p. De inhoud van de PC wordt op de stapel gezet op de manier zoals omschreven is bij de PUSH-instructie. Daarna komt het door p te specificeren adres in het lagere orde-deel van de PC. Het hogere orde-deel van de PC wordt geladen met nul.

p kan zijn: 00H, 08H, 10H, 18H, 20H, 28H, 30H of 38H

1 1 < t > 1 1 1	t= 0 0 0	p=00H	C7
	t= 0 0 1	p=08H	CF
	t= 0 1 0	p=10H	D7
	t= 0 1 1	p=18H	DF
	t= 1 0 0	p=20H	E7
	t= 1 0 1	p=28H	EF
	t= 1 1 0	p=30H	F7
	t= 1 1 1	p=38H	FF

Vlaggen niet beïnvloed.

3 M-cycli, 11(5,3,3) T-cycli. Gemodificeerde pagina 0 adressering.

SBC A,s (subtract with carry)

Trekt de door s te specificeren operand en de carry af van de inhoud van de accumulator. Het resultaat komt in de accumulator.

s kan zijn: r, n, (HL), (IX+d) of (IY+d)

r	1 0 0 1 1 < r >	r=A 1 1 1	9F
		r=B 0 0 0	98
		r=C 0 0 1	99
		r=D 0 1 0	9A
		r=E 0 1 1	9B
		r=H 1 0 0	9C
		r=L 1 0 1	9D
n	1 1 0 1 1 1 1 0		DE
	< - - -n- - - >		data

SBC A,s (vervolg)

(HL)	1 0 0 1 1 1 1 0	9E
(IX+d)	1 1 0 1 1 1 0 1	DD
	1 0 0 1 1 1 1 0	9E
	< - - -d- - - >	verplaatsing (twee-complement)
(IY+d)	1 1 1 1 1 1 0 1	FD
	1 0 0 1 1 1 1 0	9E
	< - - -d- - - >	verplaatsing (twee-complement)

carry flag	:	geset bij een borrow, anders gereset
zero flag	:	geset als het resultaat nul is, anders gereset
parity/overflow	:	geset door overflow, anders gereset
sign flag	:	geset als het resultaat negatief is, anders gereset
subtract flag	:	geset
half carry flag	:	geset bij een borrow van bit 4, anders gereset

	M-cycli	T-cycli	adresseringswijze
r	1	4	register
n	2	7(4,3)	onmiddellijk
(HL)	2	7(4,3)	register indirect
(IX+d)	5	19(4,4,3,5,3)	geïndexeerd
(IY+d)	5	19(4,4,3,5,3)	geïndexeerd

SBC HL,ss (subtract with carry)

Trekt de inhoud van het door ss te specificeren registerpaar en de carry af van de inhoud van registerpaar HL. Het resultaat komt in registerpaar HL.

ss kan zijn: BC, DE, HL of SP

```
        1 1 1 0 1 1 0 1   ED
        0 1 s s 0 0 1 0   ss=BC   0 0   42
                          ss=DE   0 1   52
                          ss=HL   1 0   62
                          ss=SP   1 1   72
```

carry flag : geset bij een borrow, anders gereset
zero flag : geset als het resultaat nul is, anders gereset
parity/overflow : geset door overflow, anders gereset
sign flag : geset als het resultaat negatief is, anders gereset
subtract flag : geset
half carry flag : geset bij een borrow van bit 12, anders gereset

4 M-cycli, 15(4,4,4,3) T-cycli.

SCF (set carry flag)

Set de carry-vlag.

```
        0 0 1 1 0 1 1 1   37
```

carry flag : geset
zero flag : niet beïnvloed
parity/overflow : niet beïnvloed
sign flag : niet beïnvloed
subtract flag : gereset
half carry flag : gereset

1 M-cyclus, 4 T-cycli.

SET b,m

Set het door b te specificeren bit van de door m te specificeren operand.
m kan zijn: r, (HL), (IX+d) of (IY+d)
bit 0 is het laagste bit.

```
r         1 1 0 0 1 0 1 1   CB
          1 1 < b > < r >   r=A   1 1 1
                            r=B   0 0 0
                            r=C   0 0 1
                            r=D   0 1 0
                            r=E   0 1 1
                            r=H   1 0 0
                            r=L   1 0 1

(HL)      1 1 0 0 1 0 1 1   CB    bit 0=   0 0 0
          1 1 < b > 1 1 0         bit 1=   0 0 1
                                  bit 2=   0 1 0
                                  bit 3=   0 1 1
                                  bit 4=   1 0 0
                                  bit 5=   1 0 1
                                  bit 6=   1 1 0
(IX+d)    1 1 0 1 1 1 0 1   DD    bit 7=   1 1 1
          1 1 0 0 1 0 1 1   CB
          < - - -d- - - >   verplaatsing (twee-complement)
          1 1 < b > 1 1 0

(IY+d)    1 1 1 1 1 1 0 1   FD
          1 1 0 0 1 0 1 1   CB
          < - - -d- - - >   verplaatsing (twee-complement)
          1 1 < b > 1 1 0
```

bit	A	B	C	D	E	H	L	(HL) en (IX+d) en (IY+d)
0	C7	C0	C1	C2	C3	C4	C5	C6
1	CF	C8	C9	CA	CB	CC	CD	CE
2	D7	D0	D1	D2	D3	D4	D5	D6
3	DF	D8	D9	DA	DB	DC	DD	DE
4	E7	E0	E1	E2	E3	E4	E5	E6
5	EF	E8	E9	EA	EB	EC	ED	EE
6	F7	F0	F1	F2	F3	F4	F5	F6
7	FF	F8	F9	FA	FB	FC	FD	FE

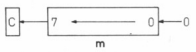
SET b,m (vervolg)

Vlaggen niet beïnvloed.

	M-cycli	T-cycli	adresseringswijze
r	2	8(4,4)	register
(HL)	4	15(4,4,4,3)	register indirect
(IX+d)	6	23(4,4,3,5,4,3)	geïndexeerd
(IY+d)	6	23(4,4,3,5,4,3)	geïndexeerd

SLA m (shift left arithmetic)

Verschuift de door m te specificeren operand een bit naar links. De inhoud van bit 7 komt in de carry en de inhoud van bit 0 wordt 0. Bit 0 is het laagste bit.



m kan zijn: r, (HL), (IX+d) of (IY+d)

r	1 1 0 0 1 0 1 1	CB	
	0 0 1 0 0 < r >	r=A 1 1 1	27
		r=B 0 0 0	20
		r=C 0 0 1	21
		r=D 0 1 0	22
		r=E 0 1 1	23
		r=H 1 0 0	24
		r=L 1 0 1	25
(HL)	1 1 0 0 1 0 1 1	CB	
	0 0 1 0 0 1 1 0	26	
(IX+d)	1 1 0 1 1 1 0 1	DD	
	1 1 0 0 1 0 1 1	CB	
	< - - -d- - - >	verplaatsing (twee-complement)	
	0 0 1 0 0 1 1 0	26	
(IY+d)	1 1 1 1 1 1 0 1	FD	
	1 1 0 0 1 0 1 1	CB	

SLA m (vervolg)

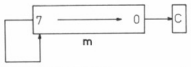
< - - -d- - - >	verplaatsing (twee-complement)
0 0 1 0 0 1 1 0	26

carry flag : voormalige inhoud bit 7
zero flag : geset als het resultaat nul is, anders gereset
parity/overflow : geset door even pariteit, anders gereset
sign flag : geset als het resultaat negatief is, anders gereset
subtract flag : gereset
half carry flag : gereset

	M-cycli	T-cycli	adresseringswijze
r	2	8(4,4)	register
(HL)	4	15(4,4,4,3)	register indirect
(IX+d)	6	23(4,4,3,5,4,3)	geïndexeerd
(IY+d)	6	23(4,4,3,5,4,3)	geïndexeerd

SRA m (shift right arithmetic)

Verschuift de door m te specificeren operand een bit naar rechts. De inhoud van bit 0 komt in de carry en de inhoud van bit 7 blijft gelijk. Bit 0 is het laagste bit.



m kan zijn: r, (HL), (IX+d) of (IY+d)

r	1 1 0 0 1 0 1 1	CB	
	0 0 1 0 1 < r >	r=A 1 1 1	2F
		r=B 0 0 0	28
		r=C 0 0 1	29
		r=D 0 1 0	2A
		r=E 0 1 1	2B
		r=H 1 0 0	2C
		r=L 1 0 1	2D

SRA m (vervolg)

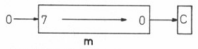
(HL)	1 1 0 0 1 0 1 1	CB
	0 0 1 0 1 1 1 0	2E
(IX+d)	1 1 0 1 1 1 0 1	DD
	1 1 0 0 1 0 1 1	CB
	< - - -d- - - >	verplaatsing (twee-complement)
	0 0 1 0 1 1 1 0	2E
(IY+d)	1 1 1 1 1 1 0 1	FD
	1 1 0 0 1 0 1 1	CB
	< - - -d- - - >	verplaatsing (twee-complement)
	0 0 1 0 1 1 1 0	2E

carry flag : voormalige inhoud bit 0
zero flag : geset als het resultaat nul is, anders gereset
parity/overflow : geset door even pariteit, anders gereset
sign flag : geset als het resultaat negatief is, anders gereset
subtract flag : gereset
half carry flag : gereset

	M-cycli	T-cycli	adresseringswijze
r	2	8(4,4)	register
(HL)	4	15(4,4,4,3)	register indirect
(IX+d)	6	23(4,4,3,5,4,3)	geïndexeerd
(IY+d)	6	23(4,4,3,5,4,3)	geïndexeerd

SRL m (shift right logical)

Verschuift de door m te specificeren operand een bit naar rechts. De inhoud van bit 0 komt in de carry en de inhoud van bit 7 wordt nul. Bit 0 is het laagste bit.



m

m kan zijn: r, (HL), (IX+d) of (IY+d)

SRL m (vervolg)

r	1 1 0 0 1 0 1 1	CB	
	0 0 1 1 1 < r >	r=A	1 1 1 3F
		r=B	0 0 0 38
		r=C	0 0 1 39
		r=D	0 1 0 3A
		r=E	0 1 1 3B
		r=H	1 0 0 3C
		r=L	1 0 1 3D
(HL)	1 1 0 0 1 0 1 1	CB	
	0 0 1 1 1 1 1 0	3E	
(IX+d)	1 1 0 1 1 1 0 1	DD	
	1 1 0 0 1 0 1 1	CB	
	< - - -d- - - >	verplaatsing (twee-complement)	
	0 0 1 1 1 1 1 0	3E	
(IY+d)	1 1 1 1 1 1 0 1	FD	
	1 1 0 0 1 0 1 1	CB	
	< - - -d- - - >	verplaatsing (twee-complement)	
	0 0 1 1 1 1 1 0	3E	

carry flag : voormalige inhoud bit 0
zero flag : geset als het resultaat nul is, anders gereset
parity/overflow : geset door even pariteit, anders gereset
sign flag : gereset
subtract flag : gereset
half carry flag : gereset

	M-cycli	T-cycli	adresseringswijze
r	2	8(4,4)	register
(HL)	4	15(4,4,4,3)	register indirect
(IX+d)	6	23(4,4,3,5,4,3)	geïndexeerd
(IY+d)	6	23(4,4,3,5,4,3)	geïndexeerd

SUB s (subtract)

Trekt de door s te specificeren operand af van de inhoud van de accumulator. Het resultaat komt in de accumulator.
s kan zijn: r, n, (HL), (IX+d) of (IY+d)

r	1 0 0 1 0 < r >	r=A 1 1 1	97
		r=B 0 0 0	90
		r=C 0 0 1	91
		r=D 0 1 0	92
		r=E 0 1 1	93
		r=H 1 0 0	94
		r=L 1 0 1	95
n	1 1 0 1 0 1 1 0		D6
	< - - -n- - - >		data
(HL)	1 0 0 1 0 1 1 0		96
(IX+d)	1 1 0 1 1 1 0 1		DD
	1 0 0 1 0 1 1 0		96
	< - - -d- - - >		verplaatsing (twee-complement)
(IY+d)	1 1 1 1 1 1 0 1		FD
	1 0 0 1 0 1 1 0		96
	< - - -d- - - >		verplaatsing (twee-complement)

carry flag : geset door een borrow, anders gereset
zero flag : geset als het resultaat nul is, anders gereset
parity/overflow : geset door overflow, anders gereset
sign flag : geset als het resultaat negatief is, anders gereset
subtract flag : geset
half carry flag : geset door een borrow van bit 4, anders gereset

	M-cycli	T-cycli	adresseringswijze
r	1	4	register
n	2	7(4,3)	onmiddellijk
(HL)	2	7(4,3)	register indirect
(IX+d)	5	19(4,4,3,5,3)	geïndexeerd
(IY+d)	5	19(4,4,3,5,3)	geïndexeerd

XOR s (exclusive or)

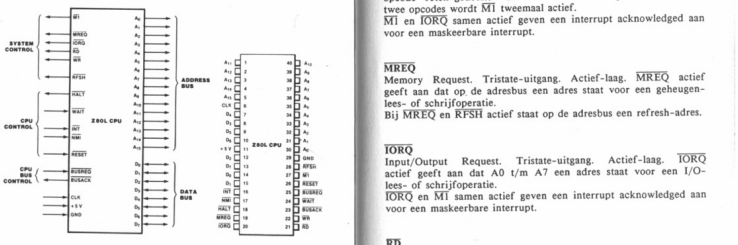
Geeft een exclusieve OF van de door s te specificeren operand en de inhoud van de accumulator. Het resultaat komt in de accumulator.
s kan zijn: r, n, (HL), (IX+d) of (IY+d)

r	1 0 1 0 1 < r >	r=A 1 1 1	AF
		r=B 0 0 0	A8
		r=C 0 0 1	A9
		r=D 0 1 0	AA
		r=E 0 1 1	AB
		r=H 1 0 0	AC
		r=L 1 0 1	AD
n	1 1 1 0 1 1 1 0		EE
	< - - -n- - - >		data
(HL)	1 0 1 0 1 1 1 0		AE
(IX+d)	1 1 0 1 1 1 0 1		DD
	1 0 1 0 1 1 1 0		AE
	< - - -d- - - >		verplaatsing (twee-complement)
(IY+d)	1 1 1 1 1 1 0 1		FD
	1 0 1 0 1 1 1 0		AE
	< - - -d- - - >		verplaatsing (twee-complement)

carry flag : gereset
zero flag : geset als het resultaat nul is, anders gereset
parity/overflow : geset door even pariteit, anders gereset
sign flag : geset als het resultaat negatief is, anders gereset
subtract flag : gereset
half carry flag : gereset

	M-cycli	T-cycli	adresseringswijze
r	1	4	register
n	2	7(4,3)	onmiddellijk
(HL)	2	7(4,3)	register indirect
(IX+d)	5	19(4,4,3,5,3)	geïndexeerd
(IY+d)	5	19(4,4,3,5,3)	geïndexeerd

9. Aansluitpennen



9.1. Pinaansluitingen.

A0 t/m A15
Adresbus. Tristate-uitgang. Actief-hoog. In totaal kunnen 2^{16}=65536 ofwel 64 Kbyte geheugenadressen worden geselecteerd. Het adresseren van I/O-apparaten gaat via A0 t/m A7, waarmee 2^{8}=256 input- of output-poorten kunnen worden geselecteerd. Op A8 t/m A15 staat dan, afhankelijk van de instructie, de inhoud van register A of B.
Tijdens de laatste twee klokcycli van een M-cyclus, de eerste

machinecyclus van een instructie ofwel de opcode fetch, bevatten A0 t/m A7 de inhoud van refresh register R en A8 t/m A15 de inhoud van register I.

D0 t/m D7
Databus. Tristate ingang/uitgang. Actief-hoog. Tweerichtings databus voor uitwisseling data met geheugen en I/O-apparaten.

$\overline{M1}$
Machinecyclus 1. Uitgang. Actief-laag. $\overline{M1}$ is actief tijdens het opcode fetch-gedeelte van een instructie. Bij instructies met twee opcodes wordt $\overline{M1}$ tweemaal actief.
$\overline{M1}$ en $\overline{IORQ}$ samen actief geven een interrupt acknowledged aan voor een maskeerbare interrupt.

$\overline{MREQ}$
Memory Request. Tristate-uitgang. Actief-laag. $\overline{MREQ}$ actief geeft aan dat op de adresbus een adres staat voor een geheugen-lees- of schrijfoperatie.
Bij $\overline{MREQ}$ en $\overline{RFSH}$ actief staat op de adresbus een refresh-adres.

$\overline{IORQ}$
Input/Output Request. Tristate-uitgang. Actief-laag. $\overline{IORQ}$ actief geeft aan dat A0 t/m A7 een adres staat voor een I/O-lees- of schrijfoperatie.
$\overline{IORQ}$ en $\overline{M1}$ samen actief geven een interrupt acknowledged aan voor een maskeerbare interrupt.

$\overline{RD}$
Read. Tristate-uitgang. Actief-laag. $\overline{RD}$ geeft aan dat de Z80 wil lezen in I/O of geheugen.

$\overline{WR}$
Write. Tristate-uitgang. Actief-laag. $\overline{WR}$ geeft aan dat de Z80 de inhoud van de databus naar I/O of geheugen wil schrijven.

$\overline{RFSH}$
Refresh. Uitgang. Actief-laag. $\overline{RFSH}$ geeft samen met $\overline{MREQ}$ aan

dat op de adresbus een refresh-adres staat voor dynamische RAM. (A0 t/m A7 inhoud van register R; A8 t/m A15 inhoud van register I.)

HALT
Halt-toestand. Uitgang. Actief-laag. HALT actief geeft aan dat de Z80 de software-instructie Halt ontvangen heeft en wacht op een niet maskeerbare interrupt, een maskeerbare (voor zover toegestaan) of een reset. In de tussentijd voert de Z80 NOP-instructies uit zodat de refresh van dynamisch geheugen doorgaat.

WAIT
Ingang. Actief-laag. Met WAIT actief geeft geheugen of I/O als reactie op een MREQ RD/WR of IORQ RD/WR aan nog niet klaar te zijn voor een lees- of schrijfoperatie. De Z80 wacht tot WAIT inactief wordt.

INT
Interrupt aanvraag. Ingang. Actief-laag. Door INT actief te maken kan een I/O-apparaat een interrupt aanvragen. De Z80 bekijkt de toestand van de INT-aansluiting tegen het einde van elke instructie. De interrupt wordt geaccepteerd als interrupt-flipflop, IFF1 door de software is geset (EI) en NMI en BUSRQ niet actief zijn.
Ten teken dat de interrupt wordt geaccepteerd maakt de Z80 IORQ en M1 actief.

NMI
Non Maskable Interrupt-aanvraag. Ingang. Een negatieve flank op de NMI-ingang triggert een interne NMI-flipflop. De toestand daarvan wordt tegen het eind van elke instructie bekeken. Tenzij BUSRQ actief is wordt de interrupt altijd geaccepteerd.

RESET
Ingang. Actief laag. De ingang moet minstens drie klokcycli actief zijn om de Z80 te resetten. Resultaat is dat de interrupt-mode, de inhoud van de PC en de registers I en R nul

worden. De interrupt-flipflops IFF1 en IFF2 worden gereset. Tijdens de reset zijn adres- en databus in de hoge impedantie-toestand en alle controle-uitgangen zijn inactief.
Niet voorkomend in Zilog-specificaties en daarom misschien niet absoluut betrouwbaar is een reset waarbij de ingang laag is tijdens een positieve flank van de klok. Hierdoor wordt alleen de inhoud van de PC nul.

BUSRQ
Bus Request. Ingang. Actief-laag. De Z80 bekijkt de toestand van de BUSRQ-aansluiting tegen het einde van elke machinecyclus. Is BUSRQ actief dan zet de Z80 aan het einde van de machine-cyclus de adresbus, databus en alle tristate-controle-uitgangen in een toestand van hoge impedantie. Deze toestand blijft gehandhaafd tot BUSRQ weer inactief is.

BUSAK
Bus Acknowledge. Uitgang. Actief-laag. Door BUSAK actief te maken geeft de Z80 aan dat in antwoord op een BUSRQ adresbus, databus en tristate-controle-uitgangen in toestand van hoge impedantie zijn gezet.
Het signaal blijft actief tot de genoemde bussen en tristate-controle-uitgangen weer in de normale toestand staan.

CLOCK
Ingang. Enkelfasige TTL-klok. Maximale frequentie voor de diverse typen:

Z80	2,5 MHz
Z80A	4 MHz
Z80B	6 MHz
Z80H	8 MHz
Z80L	(low power) versies voor 1 MHz en 2,5 MHz

+5 V en GND
Voedingsspanning: Vcc=5 V $\pm$ 5%. Hierna volgen de DC-karakteristieken.

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- S = 0°C to +70° C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- E = -40°C to +85°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- M = -55°C to +125°C, +4.5 V ≤ V_{CC} ≤ +5.5 V

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6	V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL}=1.8 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH}=-250 uA
I_{CC}	Power Supply Current Z80 Z80A Z80B		150[1] 200[2] 200	 mA mA mA	
I_{LI}	Input Leakage Current		10	µA	V_{IN}=0 to V_{CC}
I_{LO}	3-State Output Leakage Current in Float	-10	10[3]	µA	V_{OUT}=0.4 to V_{CC}

1. For military grade parts I_{CC} is 200 mA.
2. Typical rate for Z80A is 90 mA
3. A_{15}-A_0, D_7-D_0, $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$.

Symbol	Parameter	Min	Max	Unit	Note
C_{CLOCK}	Clock Capacitance		35	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance		5	pF	
C_{OUT}	Output capacitance		10	pF	

T_A =25° C, f=1 MHz

Afb. 9.2. DC-karakteristieken van de Z80, Z80A en Z80B.

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- S = 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6	V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL}=1.8 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH}=-250 uA
I_{LI}	Input Leakage Current		10	µA	V_{IN}=0 to V_{CC}
I_{LO}	3-State Output Leakage		± 10[1]	µA	V_{OUT}=0.4 to V_{CC}
I_{CC}	Power Supply Current				

	Temperature				
	0°C		25°C	70°C	
Frequency	Max	Max	Typical	Max	Unit
Z8300-1 (1.0 MHz)	30	26	15	20	mA
Z8300-3 (2.5 MHz)	45	42	25	35	mA

1. A_{15}-A_0, D_7-D_0, $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$.

Symbol	Parameter	Min	Max	Unit	Note
C_{CLOCK}	Clock Capacitance		35	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance		5	pF	
C_{OUT}	Output Capacitance		10	pF	

T_A=25° C, f=1 MHz

Afb. 9.3. DC-karakteristieken van de Z80 low power-versies.

10. Signalen op de bussen

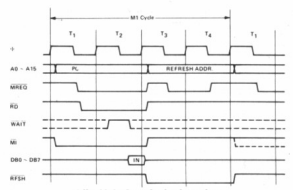
Voor de verbinding tussen de Z80 en de overige delen van een systeem is de timing van hetgeen er tijdens de volgende machine-cycli aan signalen op de bussen verschijnt van belang.

Instructie opcode fetch (M1-cyclus)
Geheugenlees- of schrijfcyclus
I/O-lees- of schrijfcyclus
Busrequest/uitvoeringcyclus
Maskeerbare interrupt request/uitvoeringcyclus
Niet maskeerbare interrupt request uitvoeringcyclus
Verlaten van de halttoestand-cyclus

Voor elke cyclus worden de signalen besproken en qua timing gerefereerd aan de systeemklok.

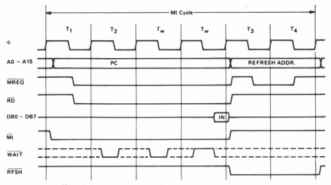
INSTRUCTIE OPCODE FETCH (M1-CYCLUS)

Elke instructie begint met het halen van een opcode-byte uit het geheugen tijdens de opcode fetch of $\overline{M1}$-cyclus. Bij instructies met twee opcodes zijn er twee opeenvolgende $\overline{M1}$-cycli. De Z80 plaatst de inhoud van de PC op de adresbus en maakt $\overline{M1}$ actief. Na een halve T-cyclus wordt $\overline{MREQ}$ actief. De inhoud van de adresbus is dan stabiel. Meteen na $\overline{MREQ}$ wordt $\overline{RD}$ actief. Tijdens de neergaande flank van T2 bekijkt de Z80 de toestand van de $\overline{WAIT}$-aansluiting. Is $\overline{WAIT}$ niet actief dan heeft het geheugen de inhoud van het adres op de databus geplaatst. De Z80 leest de data van de adresbus tijdens de opgaande flank van T3. Na de opgaande flank van T3 worden $\overline{MREQ}$ en $\overline{RD}$ inactief. Daarna ook het $\overline{M1}$-signaal.



Afb. 10.1. Opcode fetch-cyclus.

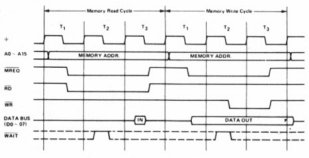
De Z80 begint met het decoderen van de instructie. Meteen na het inactief worden van $\overline{M1}$ plaatst de Z80 de inhoud van de registers I en R op de adresbus en maakt $\overline{RFSH}$ actief. De inhoud van bit 0 t/m bit 6 van het R-register wordt na elke refresh-operatie verhoogd. Bit 7 kan programmatisch 1 of 0 worden gemaakt door LD R,A. Het refresh-adres is slechts stabiel als $\overline{MREQ}$ actief is. $\overline{RFSH}$ moet daarom met $\overline{MREQ}$ worden gebruikt voor het opwekken van signalen bij de refresh van dynamische RAM.



Afb. 10.2. Opcode fetch-cyclus met wait states.

Wat betreft refresh tijdens blokverplaats- of blokvergelijk-instructies: deze instructies herhalen zichzelf na verplaatsing of vergelijking van 1 byte door de PC met 2 te verlagen, waardoor deze weer naar het begin van de instructie wijst. Na elke verplaatsing of vergelijking van 1 byte vinden dus 2 refreshes plaats.
Als de Z80 zich na een busrequest van de bussen heeft afgesloten wordt geen refresh uitgevoerd.
Is het $\overline{\text{WAIT}}$-signaal tijdens de neergaande flank van T2 wel actief dan is de inhoud van de aangewezen geheugenlocatie nog niet beschikbaar op de databus. De Z80 last een extra T-cyclus in: Tw ofwel een wait state. Op de neergaande flank van Tw wordt opnieuw de toestand van de $\overline{\text{WAIT}}$-aansluiting bekeken en indien nodig last de Z80 weer een wait state in. Op deze manier past de Z80 zich aan de snelheid van het geheugen aan. Het direct na elkaar voorkomen van een groot aantal wait states kan de refresh van dynamische RAM in gevaar brengen.

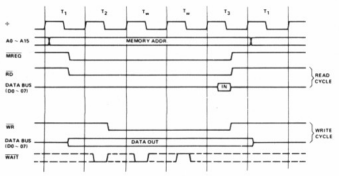
GEHEUGENLEES/SCHRIJFCYCLUS



Afb. 10.3. Geheugenlees/schrijfcyclus.

De Z80 plaatst het adres van de geheugenlocatie op de adresbus en maakt $\overline{\text{MREQ}}$ actief als de gegevens op de adresbus stabiel zijn. Voor wat betreft de leescyclus: net als bij de $\overline{\text{M1}}$-cyclus wordt meteen na $\overline{\text{MREQ}}$ ook $\overline{\text{RD}}$ actief. De Z80 bekijkt de toestand van de $\overline{\text{WAIT}}$-aansluiting tijdens de neergaande flank van T2. Is $\overline{\text{WAIT}}$ niet actief dan leest de Z80 de data in tijdens de neergaande flank van T3. Meteen hierna worden $\overline{\text{MREQ}}$ en $\overline{\text{RD}}$ inactief gemaakt. Tijdens de schrijfcyclus plaatst de Z80 data op de databus en maakt $\overline{\text{WR}}$ actief als de inhoud van de databus stabiel is. Is de toestand van de $\overline{\text{WAIT}}$-aansluiting niet actief dan last de Z80 geen wait states in.
Na de neergaande flank van T3 worden $\overline{\text{MREQ}}$ en $\overline{\text{WR}}$ inactief voor de data van de adresbus verdwijnt.
Is tijdens een lees- of schrijfoperatie de toestand van de $\overline{\text{WAIT}}$-aansluiting actief dan last de Z80 wait states in totdat, tijdens de neergaande flank van Tw, de $\overline{\text{WAIT}}$-aansluiting inactief is.

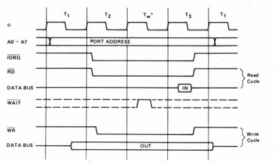
I/O-LEES/SCHRIJFCYCLUS



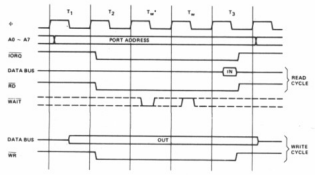
Afb. 10.4. Geheugenlees/schrijfcyclus met wait states.

De Z80 plaatst het adres van de I/O-poort op de adreslijnen A0 t/m A7 en op de lijnen A8 t/m A15, afhankelijk van de instructie, de inhoud van register A of B. De inhoud van de adresbus is stabiel als $\overline{\text{IORQ}}$ actief is.
Bij een input-instructie wordt $\overline{\text{RD}}$ meteen na $\overline{\text{IORQ}}$ actief. De Z80 bekijkt de toestand van de $\overline{\text{WAIT}}$-lijn niet zoals bij de geheugenlees/schrijfcyclus op de neergaande flank van T2 maar last auto-

matisch een wait state in om de I/O-poort genoeg tijd te geven het poortadres te decoderen.



Afb. 10.5. I/O-lees/schrijfcyclus.

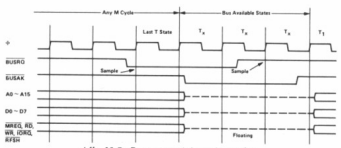


Afb. 10.6. I/O-lees/schrijf-cyclus met wait states.

Pas op de neergaande flank van Tw wordt de toestand van de WAIT-aansluiting bekeken. Is deze niet actief dan leest de Z80 het door de poort op de databus gezette byte en maakt, voordat

het adres van de adresbus verdwijnt, IORQ en RD inactief. Bij de output-instructie zet de Z80 een databyte op de databus en maakt WR actief als de databus stabiel is. Om dezelfde reden als bij de input-instructie wordt een wait state ingelast. Op de neergaande flank van Tw bekijkt de Z80 de toestand van de WAIT-aansluiting. Is WAIT niet actief dan maakt de Z80 WR en IORQ actief voor de databus en de adresbus instabiel worden. Is tijdens de lees of schrijfcyclus de toestand van de WAIT-aansluiting actief dan last de Z80 wait states in totdat op de neergaande flank van Tw het WAIT-signaal inactief is.

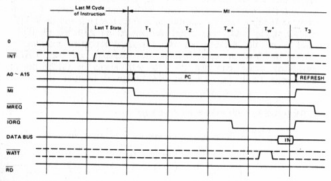
BUSREQUEST/UITVOERINGCYCLUS



Afb. 10.7. Busrequest/uitvoeringcyclus.

Tijdens de opgaande flank van de laatste T-cyclus van elke machinecyclus bekijkt de Z80 de toestand van de BUSRQ-aansluiting. Als BUSRQ actief is, zet de Z80 aan het einde van de machinecyclus de adresbus, databus en alle tristate-controle-uitgangen in een toestand van hoge impedantie. Zodra deze toestand een feit is maakt de Z80 BUSAK actief. Op de opgaande flank van elke T-cyclus bekijkt de Z80 vervolgens of BUSRQ nog steeds actief is. Zodra dat niet het geval is, worden adresbus, databus en tristate controle-uitgangen in de normale toestand gezet. BUSAK wordt inactief voor de bussen instabiel worden. De Z80 vervolgt het uitvoeren van het programma. Tijdens de busrequest worden maskeerbare en niet maskeerbare interrupts genegeerd. De Z80 voert geen refresh uit.

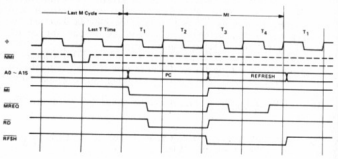
MASKEERBARE INTERRUPT REQUEST/UITVOERINGCYCLUS



Afb. 10.8. Maskeerbare interrupt request/uitvoeringcyclus.

De Z80 bekijkt de toestand van de $\overline{INT}$-aansluiting tijdens de opgaande flank van de laatste T-cyclus van elke instructie. Als $\overline{INT}$ actief is, de interrupt-flipflop IFF1 programmatisch is geset en $\overline{BUSRQ}$ noch $\overline{NMI}$ actief zijn, wordt als de instructie is uitgevoerd een speciale $\overline{M1}$-cyclus opgewekt. De Z80 plaatst de inhoud van de PC op de adresbus (wat verder geen enkele consequentie heeft) en maakt $\overline{M1}$ actief. In plaats van $\overline{MREQ}$ actief te maken zodra de inhoud van de adresbus stabiel is, maakt de Z80 $\overline{IORQ}$ actief. $\overline{M1}$ en $\overline{IORQ}$ vormen het interrupt toegestaan signaal. Twee wait states worden automatisch ingelast om een daisy chain gelegenheid te geven een vector op de databus te plaatsen.
Op de neergaande flank van de tweede wait state, bekijkt de Z80 de toestand van de $\overline{WAIT}$-aansluiting. Is deze niet actief dan wordt de inhoud van de databus gelezen waarna de Z80 een refresh uitvoert. Is de Z80 in interrupt-mode 1 dan wordt de ingelezen data genegeerd. De PC wordt tijdens de $\overline{M1}$-cyclus niet verhoogd.

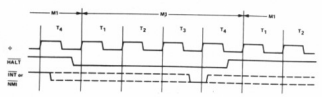
NIET MASKEERBARE INTERRUPT REQUEST/UITVOERING-CYCLUS



Afb. 10.9. Niet maskeerbare interrupt request/uitvoeringcyclus.

Een hoog naar laag overgang op de $\overline{NMI}$-aansluiting triggert een interne $\overline{NMI}$-flipflop. De Z80 bekijkt de toestand hiervan tegelijk met die van de $\overline{INT}$-aansluiting, tijdens de opgaande flank van de laatste T-cyclus van elke machinecyclus. $\overline{NMI}$ heeft een hogere prioriteit dan $\overline{INT}$. Is $\overline{NMI}$ actief en $\overline{BUSRQ}$ niet dan wordt, als de instructie is uitgevoerd, in antwoord op de $\overline{NMI}$-aanvraag een $\overline{M1}$-cyclus opgewekt. De PC wordt tijdens de $\overline{M1}$-cyclus niet verhoogd. De Z80 negeert de inhoud van de databus, zet de inhoud van PC op de stapel en plaatst 0066H in de PC.

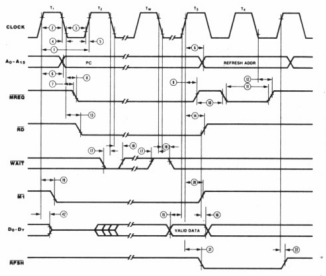
VERLATEN VAN DE HALTTOESTAND-CYCLUS



Afb. 10.10. Verlaten van de halttoestand-cyclus.

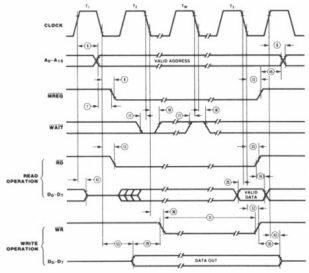
Een $\overline{\text{HALT}}$-instructie in een programma plaatst de Z80 in de halttoestand. De $\overline{\text{HALT}}$-aansluiting is actief aan het einde van de instructie. Daarna voert de Z80 NOP-instructies, ofwel $\overline{\text{M1}}$-cycli uit, tot een niet maskeerbare of een maskeerbare interrupt wordt ontvangen. De laatste alleen als maskeerbare interrupts zijn toegestaan. Tijdens de $\overline{\text{M1}}$-cycli negeert de Z80 de door het geheugen op de databus geplaatste opcode. De inhoud van de PC wordt tijdens de $\overline{\text{M1}}$-cycli niet verhoogd,
De Z80 bekijkt de $\overline{\text{INT}}$-aansluiting en interne $\overline{\text{NMI}}$ tijdens de opgaande flank van de laatste T-cyclus van de NOP-instructie en beantwoordt de eventuele interrupt-aanvraag op de gebruikelijke manier.
De $\overline{\text{HALT}}$-aansluiting is inactief aan het einde van de laatste $\overline{\text{M1}}$-cyclus.

Op de volgende pagina's is de exacte timing van de signalen voor de diverse Z80-typen te vinden.

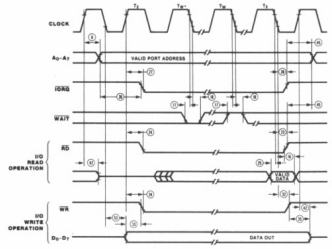


NOTE: T_W-Wait cycle added when necessary for slow ancillary devices.

Afb. 10.11. Opcode fetch-cyclus.

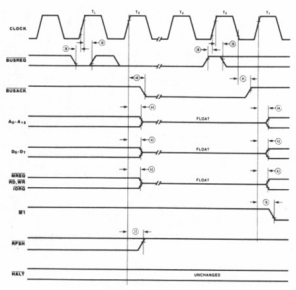


Afb. 10.12. Lees/schrijfcyclus.



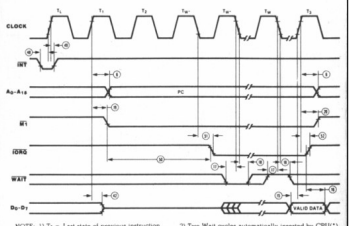
NOTE: T_{W^*} = One Wait cycle automatically inserted by CPU.

Afb. 10.13. I/O-lees/schrijfcyclus.



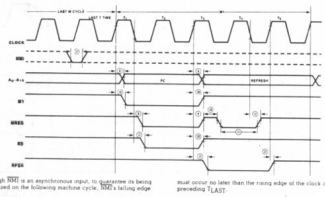
NOTE T_L = Last state of any M cycle. T_X = An arbitrary clock cycle used by requesting device.

Afb. 10.14. Busrequest/uitvoeringcyclus.



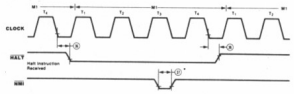
NOTE: 1) T_L = Last state of previous instruction. 2) Two Wait cycles automatically inserted by CPU(*).

Afb. 10.15. Maskeerbare interrupt request/uitvoeringcyclus.



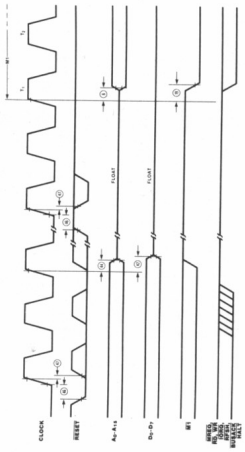
though NMI is an asynchronous input, to guarantee its being recognized on the following machine cycle, NMI's falling edge must occur no later than the rising edge of the clock cycle preceding T_LAST.

Afb. 10.16. Niet maskeerbare interrupt request/uitvoering-cyclus.



NOTE: INT will also force a Halt exit. *See note, Figure 9.

Afb. 10.17. Verlaten van de halttoestand-cyclus.



Afb. 10.18. Reset cyclus.

Number	Symbol	Parameter	Z80 CPU Min	Z80 CPU Max	Z80A CPU Min	Z80A CPU Max	Z80B CPU	Z80H CPU*
1	TcC	Clock Cycle Time	400*	250*	165*	125*		
2	TwCh	Clock Pulse Width (High)	180*	110*	65*	55*		
3	TwCl	Clock Pulse Width (Low)	180	2000	110	2000	65	2000	55	2000
4	TfC	Clock Fall Time	-	30	-	30	-	20	-	10
5	TrC	Clock Rise Time	-	30	-	30	-	20	-	10
6	TdC(A)	Clock ↑ to Address Valid Delay	-	145	-	110	-	90	-	80
7	TdA(MREQf)	Address Valid to MREQ ↓ Delay	125*	-	65*	-	35*	-	20*	-
8	TdC(MREQf)	Clock ↓ to MREQ ↓ Delay	-	100	-	85	-	70	-	60
9	TdC(MREQr)	Clock ↑ to MREQ ↑ Delay	-	100	-	85	-	70	-	60
10	TwMREQh	MREQ Pulse Width (High)	170*		110*		65*		45*	-
11	TwMREQl	MREQ Pulse Width (Low)	360*	-	220*	-	135*	-	100*	-
12	TdC(MREQr)	Clock ↓ to MREQ ↑ Delay	-	100	-	85	-	70	-	60
13	TdC(RDf)	Clock ↓ to RD ↓ Delay	-	130	-	95	-	80	-	70
14	TdC(RDr)	Clock ↑ to RD ↑ Delay	-	100	-	85	-	70	-	60
15	TsD(CR)	Data Setup Time to Clock ↑	50		35		30		30	-
16	ThD(RDr)	Data Hold Time to RD ↑	-	0	-	0	-	0	-	0
17	TsWAIT(Cf)	WAIT Setup Time to Clock ↓	70	-	70	-	60	-	50	-
18	ThWAIT(C)	WAIT Hold Time after Clock ↓	-	0	-	0	-	0	-	0
19	TdC(MIf)	Clock ↑ to M1 ↓ Delay	-	130	-	100	-	80	-	70
20	TdC(M1r)	Clock ↑ to M1 ↑ Delay		130		100		80	-	70
21	TdC(RFSHf)	Clock ↑ to RFSH ↓ Delay	-	180	-	130	-	110	-	95
22	TdC(RFSHr)	Clock ↑ to RFSH ↑ Delay	-	150	-	120	-	100	-	85
23	TdCf(RDr)	Clock ↓ to RD ↑ Delay	-	110	-	85	-	70	-	60
24	TdCr(RDf)	Clock ↑ to RD ↓ Delay	-	100	-	85	-	70	-	60
25	TsD(Cf)	Data Setup to Clock ↓ during M_2, M_3, M_4 or M_5 Cycle	60		50		40		30	-
26	TdA(IORQf)	Addres Stable Prior to IORQ ↓	320*	-	180*	-	110*	-	75*	-
27	TdC(IORQf)	Clock ↑ to IORQ ↓ Delay	-	90	-	75	-	65	-	55
28	TdC(IORQr)	Clock ↓ to IORQ ↑ Delay	-	110	-	85	-	70	-	60
29	TdD(WRf)	Data Stable Prior to WR ↓	190*	-	80*	-	25*	-	5*	-
30	TdC(WRf)	Clock ↓ to WR ↓ Delay		90		80		70	-	60
31	TwWR	WR Pulse Width	360*	-	220*	-	135*	-	100*	-
32	TdC(WRr)	Clock ↓ to WR ↑ Delay	-	100	-	80	-	70	-	60
33	TdD(WRf)	Data Stable prior to WR ↓	20*	-	-10*	-	-55*	-	55*	-
34	TdCr(WRf)	Clock ↑ to WR ↓ Delay	-	80	-	65	-	60	-	55
35	TdWR(D)	Data Stable from WR ↑	120*		60*		30*		15*	-
36	TdC(HALT)	Clock ↓ to HALT ↑ or ↓	-	300	-	300	-	260	-	225
37	TwNMI	NMI Pulse Width	80	-	80	-	70	-	60*	-
38	TsBUSREQ(Cr)	BUSREQ Setup ime to Clock ↑	80	-	50	-	50	-	40	-

Afb. 10.19.

Numb.	Symbol	Parameter	Z80 CPU Min	Z80 CPU Max	Z80A CPU Min	Z80A CPU Max	Z80B CPU Min	Z80B CPU Max	Z80H CPU* Min	Z80H CPU* Max
39	ThBUSREQ(Cr)	BUSREQ Hold Time after Clock ↑	0		0		0		0	
40	TdC(BUSACKf)	Clock ↑ to BUSACK ↓ Delay		120		100		90		80
41	TdC(BUSACKr)	Clock ↓ to BUSACK ↑ Delay	-	110	-	100	-	90	-	80
42	TdCr(Dz)	Clock ↑ to Data Float Delay	-	90	-	90	-	80	-	70
43	TdCr(CTz)	Clock ↑ to Control Outputs Float Delay (MREQ, IORQ, RD, and WR)	-	110	-	80	-	70	-	60
44	TdC(Az)	Clock ↑ to Address Float Delay	-	110	-	90	-	80	-	70
45	TdCTr(Az)	MREQ↓, IORQ↓, RD↓, and WR↓ to Address Hold Time	160*		80*		35*		20*	-
46	TsRESET(Cr)	RESET to Clock ↑ Setup Time	90	-	60	-	60	-	45	-
47	ThRESET(Cr)	RESET to Clock ↑ Hold Time	-	0	-	0	-	0	-	0
48	TsINT(Cr)	INT to Clock ↑ Setup Time	80	-	80	-	70	-	55	-
49	ThINT(Cr)	INT to Clock ↑ Hold Time	-	0	-	0	-	0	-	0
50	TdM1(IORQf)	M1 ↓ to IORQ ↓ Delay	920*	-	565*	-	365*	-	270*	-
51	TdC(IOQRQf)	Clock ↓ to IORQ ↓ Delay	-	110	-	85	-	70	-	60
52	TdC(IOQRQr)	Clock ↑ to IORQ ↑ Delay	-	100	-	85	-	70	-	60
53	TdCf(D)	Clock ↓ to Data Valid Delay	-	230	-	150	-	130	-	115

*For clock periods other than the minimum shown in the table, calculate parameters using the following expressions. Calculated values above assumed TrC + TfC = 30 ns.

† Units in nanoseconds (ns). All timings are preliminary and subject to change.

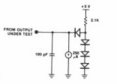
Footnotes to AC Characteristics

Numb.	Symbol	Z80	Z80A	Z80B
1	TcC	TwCh+TwCl+TrC+TfC	TwCh+TwCl+TrC+TfC	TwCh+TwCl+TrC+TfC
2	TwCh	Although static by design, TwCh of greater then 200 ns is not guaranteed	Although static by design, TwCh of greater than 200 ns is not guaranteed	Although static by design, TwCh of greater than 200 ns is not guaranteed
7	TdA(MREQf)	TwCh+TfC-75	TwCh+TfC-65	TwCh+TfC-50
10	TwMREQh	TwCh+TfC-30	TwCh+TfC-20	TwCh+TfC-20
11	TwMREQl	TcC-40	TcC-30	TcC-30
28	TdA(IOR(Qf)	TcC-80	TcC-70	TcC-55
29	TdD(WRf)	TcC-210	TcC-170	TcC-140
31	TwWR	TcC-40	TcC-30	TcC-30
33	TdD(WRf)	TwCh+TrC-180	TwCh+TrC-140	TwCh+TrC-140
35	TdWR(D)	TwCh+TrC-80	TwCh+TrC-70	TwCh+TrC-55
45	TdCTr(A)	TwCh+TrC-60	TwCh+TrC-50	TwCh+TrC-50
50	TdM1f(IORQf)	2TcC+TwCh+TfC-65	2TcC+TwCh+TfC-65	2TcC+TwCh+TfC-50

AC Test Conditions

V_{IH} = 2.0 V
V_{IL} = 0.8 V
V_{HC} = V_{CC} -0.6 V
V_{LC} = 0.45 V
V_{OH} = 2.0 V
V_{OL} = 0.8 V
FLOAT = ± 0.5 V

All ac parameters assume a load capacitance of 100 pF. Add 10 ns delay for each 50 pF increase in load up to a maximum of 200 pF for the data bus and 100 pF for address and control lines.



Afb. 10.20.

Number	Symbol	Parameter	Z8300-1 (1.0 MHz) Min (ns)	Z8300-1 (1.0 MHz) Max (ns)	Z8300-3 (2.5 MHz) Min (ns)	Z8300-3 (2.5 MHz) Max (ns)
1	TcC	Clock Cycle Time	1000*		400*	
2	TwCh	Clock Pulse Width (High)	470*		180*	
3	TwCl	Clock Pulse Width (Low)	470	2000	180	2000
4	TfC	Clock Fall Time	-	30	-	30
5	TrC	Clock Rise Time	-	30	-	30
6	TdCr(A)	Clock ↑ to Address Valid Delay	-	380	-	145
7	TdA(MREQf)	Address Valid to $\overline{MREQ}$ ↓ Delay	370*	-	125*	-
8	TdCf(MREQf)	Clock ↓ to $\overline{MREQ}$ ↓ Delay	-	260	-	100
9	TdCr(MREQr)	Clock ↑ to $\overline{MREQ}$ ↑ Delay	-	260	-	100
10	TwMREQh	$\overline{MREQ}$ Pulse Width (High)	410*		170*	
11	TwMREQl	$\overline{MREQ}$ Pulse Width (Low)	890*	-	360*	-
12	TdCf(MREQr)	Clock ↓ to $\overline{MREQ}$ ↑ Delay	-	260	-	100
13	TdCf(RDf)	Clock ↓ to $\overline{RD}$ ↓ Delay	-	340	-	130
14	TdCr(RDr)	Clock ↑ to $\overline{RD}$ ↑ Delay	-	260	-	100
15	TsDC(r)	Data Setup Time to Clock ↑	140		50	
16	ThD(RDr)	Data Hold Time to RD↑	-	0	-	0
17	TsWAIT(Cf)	$\overline{WAIT}$ Setup Time to Clock↓	190	-	70	-
18	ThWAIT(Cf)	$\overline{WAIT}$ Holt Time after Clock↓	-	0	-	0
19	TdCr(M1f)	Clock ↑ to $\overline{M1}$ ↓ Delay	-	340	-	130
20	TdCr(M1r)	Clock ↑ to $\overline{M1}$ ↑ Delay		340		130
21	TdCr(RFSHf)	Clock ↑ to $\overline{RFSH}$ ↓ Delay	-	460	-	180
22	TdCr(RFSHr)	Clock ↑ to $\overline{RFSH}$ ↑ Delay	-	390	-	150
23	TdCf(RDr)	Clock ↓ to $\overline{RD}$ ↑ Delay	-	290	-	110
24	TdCr(RDf)	Clock ↑ to $\overline{RD}$ ↓ Delay	-	260	-	100
25	TsD(Cf)	Data Setup to Clock ↓ during M_2, M_3, M_4 or M_5 Cycles	160		60	
26	TdA(IORQf)	Address Stable prior to $\overline{IORQ}$ ↓	790*	-	320*	-
27	TdCr(IORQf)	Clock ↑ to $\overline{IORQ}$ ↓ Delay	-	240	-	90
28	TdCf(IORQr)	Clock ↓ to $\overline{IORQ}$ ↑ Delay	-	290	-	110
29	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	470*	-	190*	-
30	TdCf(WRf)	Clock ↓ to $\overline{WR}$ ↓ Delay		240		90
31	TwWR	$\overline{WR}$ Pulse Width	890*	-	360*	-
32	TdCf(WRr)	Clock ↓ to $\overline{WR}$ ↑ Delay	-	260	-	100
33	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	-30*	-	30*	-
34	TdCr(WRf)	Clock ↑ to $\overline{WR}$ ↓ Delay	-	210	-	80
35	TdWR(D)	Data Stable from $\overline{WR}$ ↑	290*		130*	
36	TdCf(HALT)	Clock ↓ to $\overline{HALT}$ ↑ or ↓	-	760	-	300
37	TwNMI	$\overline{NMI}$ Pulse Width	210	-	80	-
38	TsBUSREQ(Cr)	$\overline{BUSREQ}$ Setup Time to Clock ↑	210	-	80	-

Afb. 10.21.

Number	Symbol	Parameter	Z8300-1 Min (ns)	Z8300-1 Max (ns)	Z8300-3 Min (ns)	Z8300-3 Max (ns)
39	ThBUSREQ(Cr)	$\overline{BUSREQ}$ Hold Time after Clock	0	-	0	-
40	TdCr(BUSACKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay		310		120
41	TdCf(BUSACKr)	Clock ↓ to $\overline{BUSACK}$ ↑ Delay	-	290	-	110
42	TdCr(Dz)	Clock ↑ to Data Float Delay	-	240	-	90
43	TdCr(CTz)	Clock ↑ to Control Outputs Float Delay ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$)	-	290	-	110
44	TdCr(Az)	Clock ↑ to Address Float Delay	-	290	-	110
45	TdCTr(A)	$\overline{MREQ}$↑, $\overline{IORQ}$↑, $\overline{RD}$↑, and $\overline{WR}$↑ to Address Hold Time	400*		160*	
46	TsRESET(Cr)	$\overline{RESET}$ to Clock ↑ Setup Time	240	-	90	-
47	ThRESET(Cr)	$\overline{RESET}$ to Clock ↑ Hold Time	-	0	-	0
48	TsINT(Cr)	$\overline{INT}$ to Clock ↑ Setup Time	210	-	80	-
49	ThINTr(Cr)	$\overline{INT}$ to Clock ↑ Hold Time	-	0	-	0
50	TdM1f(IORQf)	$\overline{M1}$ ↓ to $\overline{IORQ}$ ↓ Delay	2300*		920*	
51	TdCf(IORQf)	Clock ↓ to $\overline{IORQ}$ ↓ Delay	-	290	-	110
52	TdCf(IORQr)	Clock ↑ to $\overline{IORQ}$ ↑ Delay	-	260	-	100
53	TdCf(D)	Clock ↓ to Data Valid Delay	-	290	-	230

*For clock periods other than the minimums shown in the table, calculate parameters using the following expressions. Calculated value above assumed TrC=TfC=20 ns.
† All timings assume equal loading on pins with 50 pF.
Timings are preliminary and subject to change.

Footnotes to AC Characteristics

Number	Symbol	Z8300-1	Z8300-3
1	TcC	TwCh + TwCl + TrC + TfC	TwCh + TwCl + TrC + TfC
2	TwCh	Although static by design, TwCh of greater than 200 µs is not guaranteed	Although static by design, TwCh of greater than 200 µs is not guaranteed
7	TdA(MREQf)	TwCh + TfC - 200	TwCh + TfC - 75
10	TwMREQh	TwCh + TfC - 90	TwCh + TfC - 30
11	TwMREQl	TcC - 110	TcC - 30
26	TdA(IORQf)	TcC - 210	TcC - 210
29	TdD(WRf)	TcC - 540	TcC - 210
31	TwWR	TcC - 110	TcC - 40
33	TdD(WRf)	TwCl + TrC - 470	TwCl + TrC - 180
35	TdWR(D)	TwCl + TrC - 210	TwCl + TrC - 80
45	TdCTr(a)	TwCl + TrC - 110	TwCl + TrC - 40
50	TdM1f(IORQf)	2TcC + TwCh + TfC - 210	2TcC + TwCh + TfC - 80

AC Test Conditions:
V_{IH} = 2.0 V
V_{IL} = 0.8 V
V_{IHC} = V_{CC} -0.6 V
V_{ILC} = 0.45 V
V_{OH} = 2.0 V
V_{OL} = 0.8 V, FLOAT = ± 0.5 V

Afb. 10.22.

Appendix. Instructieset op volgorde van opcode

N = 8 bits
NN = 16 bits
DIS = 8 bits verplaatsing (displacement)
IND = 8 bits index

De opcodes zijn als volgt verdeeld:

1 byte opcodes
2 bytes opcodes beginnend met CB
2 bytes opcodes beginnend met DD
3 bytes opcodes beginnend met DDCB
2 bytes opcodes beginnend met ED
2 bytes opcodes beginnend met FD
3 bytes opcodes beginnend met FDCB

00	NOP	0F	RRCA
01	LD BC,NN	10	DJNZ DIS
02	LD (BC),A	11	LD DE,NN
03	INC BC	12	LD (DE),A
04	INC B	13	INC DE
05	DEC B	14	INC D
06	LD B,N	15	DEC D
07	RLCA	16	LD D,N
08	EX AF,AF'	17	RLA
09	ADD HL,BC	18	JR DIS
0A	LD A,(BC)	19	ADD HL,DE
0B	DEC BC	1A	LD A,(DE)
0C	INC C	1B	DEC DE
0D	DEC C	1C	INC E
0E	LD C,N	1D	DEC E

1E	LD E,N	49	LD C,C
1F	RRA	4A	LD C,D
20	JR NZ,DIS	4B	LD C,E
21	LD HL,NN	4C	LD C,H
22	LD (NN),HL	4D	LD C,L
23	INC HL	4E	LD C,(HL)
24	INC H	4F	LD C,A
25	DEC H	50	LD D,B
26	LD H,N	51	LD D,C
27	DAA	52	LD D,D
28	JR Z,DIS	53	LD D,E
29	ADD HL,HL	54	LD D,H
2A	LD HL,(NN)	55	LD D,L
2B	DEC HL	56	LD D,(HL)
2C	INC L	57	LD D,A
2D	DEC L	58	LD E,B
2E	LD L,N	59	LD E,C
2F	CPL	5A	LD E,D
30	JR NC,DIS	5B	LD E,E
31	LD SP,NN	5C	LD E,H
32	LD (NN),A	5D	LD E,L
33	INC SP	5E	LD E,(HL)
34	INC (HL)	5F	LD E,A
35	DEC (HL)	60	LD H,B
36	LD (HL),N	61	LD H,C
37	SCF	62	LD H,D
38	JR C,DIS	63	LD H,E
39	ADD HL, SP	64	LD H,H
3A	LD A,(NN)	65	LD H,L
3B	DEC SP	66	LD H,(HL)
3C	INC A	67	LD H,A
3D	DEC A	68	LD L,B
3E	LD A,N	69	LD L,C
3F	CCF	6A	LD L,D
40	LD B,B	6B	LD L,E
41	LD B,C	6C	LD L,H
42	LD B,D	6D	LD L,L
43	LD B,E	6E	LD L,(HL)
44	LD B,H	6F	LD L,A
45	LD B,L	70	LD (HL),B
46	LD B,(HL)	71	LD (HL),C
47	LD B,A	72	LD (HL),D
48	LD C,B	73	LD (HL),E

74	LD (HL),H
75	LD (HL),L
76	HALT
77	LD (HL),A
78	LD A,B
79	LD A,C
7A	LD A,D
7B	LD A,E
7C	LD A,H
7D	LD A,L
7E	LD A,(HL)
7F	LD A,A
80	ADD A,B
81	ADD A,C
82	ADD A,D
83	ADD A,E
84	ADD A,H
85	ADD A,L
86	ADD A,(HL)
87	ADD A,A
88	ADC A,B
89	ADC A,C
8A	ADC A,D
8B	ADC A,E
8C	ADC A,H
8D	ADC A,L
8E	ADC A,(HL)
8F	ADC A,A
90	SUB B
91	SUB C
92	SUB D
93	SUB E
94	SUB H
95	SUB L
96	SUB (HL)
97	SUB A
98	SBC A,B
99	SBC A,C
9A	SBC A,D
9B	SBC A,E
9C	SBC A,H
9D	SBC H,L
9E	SBC A,(HL)
9F	SBC A,A
A0	AND B
A1	AND C
A2	AND D
A3	AND E
A4	AND H
A5	AND L
A6	AND (HL)
A7	AND A
A8	XOR B
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
AE	XOR (HL)
AF	XOR A
B0	OR B
B1	OR C
B2	OR D
B3	OR E
B4	OR H
B5	OR L
B6	OR (HL)
B7	OR A
B8	CP B
B9	CP C
BA	CP D
BB	CP E
BC	CP H
BD	CP L
BE	CP (HL)
BF	CP A
C0	RET NZ
C1	POP BC
C2	JP NZ,NN
C3	JP NN
C4	CALL NZ,NN
C5	PUSH BC
C6	ADD A,N
C7	RST O
C8	RET Z
C9	RET

CA	JP Z,NN
CC	CALL Z,NN
CD	CALL NN
CE	ADC A,N
CF	RST 8H
D0	RET NC
D1	POP DE
D2	JP NC,NN
D3	OUT (N),A
D4	CALL NC,NN
D5	PUSH DE
D6	SUB N
D7	RST 10H
D8	RET C
D9	EXX
DA	JP C,NN
DB	IN A,N
DC	CALL C,NN
DE	SBC A,N
DF	RST 18H
E0	RET PO
E1	POP HL
E2	JP PO,NN
E3	EX (SP),HL
E4	CALL PO,NN
E5	PUSH HL
E6	AND N
E7	RST 20H
E8	RET PE
E9	JP (HL)
EA	JP PE,NN
EB	EX DE,HL
EC	CALL PE,NN
EE	XOR N
EF	RST 28H
F0	RET P
F1	POP AF
F2	JP P,NN
F3	DI
F4	CALL P,NN
F5	PUSH AF
F6	OR N
F7	RST 30H
F8	RET M
F9	LD SP,HL
FA	JP M,NN
FB	EI
FC	CALL M,NN
FE	CP N
FF	RST 38H
CB00	RLC B
CB01	RLC C
CB02	RLC D
CB03	RLC E
CB04	RLC H
CB05	RLC L
CB06	RLC (HL)
CB07	RLC A
CB08	RRC B
CB09	RRC C
CB0A	RRC D
CB0B	RRC E
CB0C	RRC H
CB0D	RRC L
CB0E	RRC (HL)
CB0F	RRC A
CB10	RL B
CB11	RL C
CB12	RL D
CB13	RL E
CB14	RL H
CB15	RL L
CB16	RL (HL)
CB17	RL A
CB18	RR B
CB19	RR C
CB1A	RR D
CB1B	RR E
CB1C	RR H
CB1D	RR L
CB1E	RR (HL)
CB1F	RR A
CB20	SLA B
CB21	SLA C
CB22	SLA D
CB23	SLA E

CB24 SLA H
CB25 SLA L
CB26 SLA (HL)
CB27 SLA A
CB28 SRA B
CB29 SRA C
CB2A SRA D
CB2B SRA E
CB2C SRA H
CB2D SRA L
CB2E SRA (HL)
CB2F SRA A
CB38 SRL B
CB39 SRL C
CB3A SRL D
CB3B SRL E
CB3C SRL H
CB3D SRL L
CB3E SRL (HL)
CB3F SRL A
CB40 BIT 0,B
CB41 BIT 0,C
CB42 BIT 0,D
CB43 BIT 0,E
CB44 BIT 0,H
CB45 BIT 0,L
CB46 BIT 0,(HL)
CB47 BIT 0,A
CB48 BIT 1,B
CB49 BIT 1,C
CB4A BIT 1,D
CB4B BIT 1,E
CB4C BIT 1,H
CB4D BIT 1,L
CB4E BIT 1,(HL)
CB4F BIT 1,A
CB50 BIT 2,B
CB51 BIT 2,C
CB52 BIT 2,D
CB53 BIT 2,E
CB54 BIT 2,H
CB55 BIT 2,L
CB56 BIT 2,(HL)
CB57 BIT 2,A
CB58 BIT 3,B
CB59 BIT 3,C
CB5A BIT 3,D
CB5B BIT 3,E
CB5C BIT 3,H
CB5D BIT 3,L
CB5E BIT 3,(HL)
CB5F BIT 3,A
CB60 BIT 4,B
CB61 BIT 4,C
CB62 BIT 4,D
CB63 BIT 4,E
CB64 BIT 4,H
CB65 BIT 4,L
CB66 BIT 4,(HL)
CB67 BIT 4,A
CB68 BIT 5,B
CB69 BIT 5,C
CB6A BIT 5,D
CB6B BIT 5,E
CB6C BIT 5,H
CB6D BIT 5,L
CB6E BIT 5,(HL)
CB6F BIT 5,A
CB70 BIT 6,B
CB71 BIT 6,C
CB72 BIT 6,D
CB73 BIT 6,E
CB74 BIT 6,H
CB75 BIT 6,L
CB76 BIT 6,(HL)
CB77 BIT 6,A
CB78 BIT 7,B
CB79 BIT 7,C
CB7A BIT 7,D
CB7B BIT 7,E
CB7C BIT 7,H
CB7D BIT 7,L
CB7E BIT 7,(HL)
CB7F BIT 7,A
CB80 RES 0,B
CB81 RES 0,C

CB82 RES 0,D
CB83 RES 0,E
CB84 RES 0,H
CB85 RES 0,L
CB86 RES 0,(HL)
CB87 RES 0,A
CB88 RES 1,B
CB89 RES 1,C
CB8A RES 1,D
CB8B RES 1,E
CB8C RES 1,H
CB8D RES 1,L
CB8E RES 1,(HL)
CB8F RES 1,A
CB90 RES 2,B
CB91 RES 2,C
CB92 RES 2,D
CB93 RES 2,E
CB94 RES 2,H
CB95 RES 2,L
CB96 RES 2,(HL)
CB97 RES 2,A
CB98 RES 3,B
CB99 RES 3,C
CB9A RES 3,D
CB9B RES 3,E
CB9C RES 3,H
CB9D RES 3,L
CB9E RES 3,(HL)
CB9F RES 3,A
CBA0 RES 4,B
CBA1 RES 4,C
CBA2 RES 4,D
CBA3 RES 4,E
CBA4 RES 4,H
CBA5 RES 4,L
CBA6 RES 4,(HL)
CBA7 RES 4,A
CBA8 RES 5,B
CBA9 RES 5,C
CBAA RES 5,D
CBAB RES 5,E
CBAC RES 5,H
CBAD RES 5,L
CBAE RES 5,(HL)
CBAF RES 5,A
CBB0 RES 6,B
CBB1 RES 6,C
CBB2 RES 6,D
CBB3 RES 6,E
CBB4 RES 6,H
CBB5 RES 6,L
CBB6 RES 6,(HL)
CBB7 RES 6,A
CBB8 RES 7,B
CBB9 RES 7,C
CBBA RES 7,D
CBBB RES 7,E
CBBC RES 7,H
CBBD RES 7,L
CBBE RES 7,(HL)
CBBF RES 7,A
CBC0 SET 0,B
CBC1 SET 0,C
CBC2 SET 0,D
CBC3 SET 0,E
CBC4 SET 0,H
CBC5 SET 0,L
CBC6 SET 0,(HL)
CBC7 SET 0,A
CBC8 SET 1,B
CBC9 SET 1,C
CBCA SET 1,D
CBCB SET 1,E
CBCC SET 1,H
CBCD SET 1,L
CBCE SET 1,(HL)
CBCF SET 1,A
CBD0 SET 2,B
CBD1 SET 2,C
CBD2 SET 2,D
CBD3 SET 2,E
CBD4 SET 2,H
CBD5 SET 2,L
CBD6 SET 2,(HL)
CBD7 SET 2,A

CBD8 SET 3,B
CBD9 SET 3,C
CBDA SET 3,D
CBDB SET 3,E
CBDC SET 3,H
CBDD SET 3,L
CBDE SET 3,(HL)
CBDF SET 3,A
CBE0 SET 4,B
CBE1 SET 4,C
CBE2 SET 4,D
CBE3 SET 4,E
CBE4 SET 4,H
CBE5 SET 4,L
CBE6 SET 4,(HL)
CBE7 SET 4,A
CBE8 SET 5,B
CBE9 SET 5,C
CBEA SET 5,D
CBEB SET 5,E
CBEC SET 5,H
CBED SET 5,L
CBEE SET 5,(HL)
CBEF SET 5,A
CBF0 SET 6,B
CBF1 SET 6,C
CBF2 SET 6,D
CBF3 SET 6,E
CBF4 SET 6,H
CBF5 SET 6,L
CBF6 SET 6,(HL)
CBF7 SET 6,A
CBF8 SET 7,B
CBF9 SET 7,C
CBFA SET 7,D
CBFB SET 7,E
CBFC SET 7,H
CBFD SET 7,L
CBFE SET 7,(HL)
CBFF SET 7,A
DD09 ADD IX,BC
DD19 ADD IX,DE
DD21 LD IX,NN
DD22 LD (NN),IX
DD23 INC IX
DD29 ADD IX,IX
DD2A LD IX,(NN)
DD2B DEC IX
DD34 INC (IX+IND)
DD35 DEC (IX+IND)
DD36 LD (IX+IND),N
DD39 ADD IX,SP
DD46 LD B,(IX+IND)
DD4E LD C,(IX+IND)
DD56 LD D,(IX+IND)
DD5E LD E,(IX+IND)
DD66 LD H,(IX+IND)
DD6E LD L,(IX+IND)
DD70 LD (IX+IND),B
DD71 LD (IX+IND),C
DD72 LD (IX+IND),D
DD73 LD (IX+IND),E
DD74 LD (IX+IND),H
DD75 LD (IX+IND),L
DD77 LD (IX+IND),A
DD7E LD A,(IX+IND)
DD86 ADD A,(IX+IND)
DD8E ADC A,(IX+IND)
DD96 SUB (IX+IND)
DD9E SBC A,(IX+IND)
DDA6 AND (IX+IND)
DDAE XOR (IX+IND)
DDB6 OR (IX+IND)
DDBE CP (IX+IND)
DDE1 POP IX
DDE3 EX (SP),IX
DDE5 PUSH IX
DDE9 JP (IX)
DDF9 LD SP,IX
DDCB 06 RLC (IX+IND)
DDCB 0E RRC (IX+IND)
DDCB 16 RL (IX+IND)
DDCB 1E RR (IX+IND)
DDCB 26 SLA (IX+IND)
DDCB 2E SRA (IX+IND)
DDCB 3E SRL (IX+IND)

DDCB 46 BIT 0,(IX+IND)
DDCB 4E BIT 1,(IX+IND)
DDCB 56 BIT 2,(IX+IND)
DDCB 5E BIT 3,(IX+IND)
DDCB 66 BIT 4,(IX+IND)
DDCB 6E BIT 5,(IX+IND)
DDCB 76 BIT 6,(IX+IND)
DDCB 7E BIT 7,(IX+IND)
DDCB 86 RES 0,(IX+IND)
DDCB 8E RES 1,(IX+IND)
DDCB 96 RES 2,(IX+IND)
DDCB 9E RES 3,(IX+IND)
DDCB A6 RES 4,(IX+IND)
DDCB AE RES 5,(IX+IND)
DDCB B6 RES 6,(IX+IND)
DDCB BE RES 7,(IX+IND)
DDCB C6 SET 0,(IX+IND)
DDCB CE SET 1,(IX+IND)
DDCB D6 SET 2,(IX+IND)
DDCB DE SET 3,(IX+IND)
DDCB E6 SET 4,(IX+IND)
DDCB EE SET 5,(IX+IND)
DDCB E6 SET 6,(IX+IND)
DDCB FE SET 7,(IX+IND)
ED40 IN B,(C)
ED41 OUT (C),B
ED42 SBC HL,BC
ED43 LD (NN),BC
ED44 NEG
ED45 RETN
ED46 IM 0
ED47 LD I,A
ED48 IN C,(C)
ED49 OUT (C),C
ED4A ADC HL,BC
ED4B LD BC,(NN)
ED4D RETI
ED4F LD R,A
ED50 IN D,(C)
ED51 OUT (C),D
ED52 SBC HL,DE
ED53 LD (NN),DE
ED56 IM 1
ED57 LD A,I
ED58 IN E,(C)
ED59 OUT (C),E
ED5A ADC HL,DE
ED5B LD DE,(NN)
ED5E IM 2
ED5F LD A,R
ED60 IN H,(C)
ED61 OUT (C),N
ED62 SBC HL,HL
ED63 LD (NN),HL
ED67 RRD
ED68 IN L,(C)
ED69 OUT (C),L
ED6A ADC HL,HL
ED6B LD HL,(NN)
ED6F RLD
ED72 SBC HL,SP
ED73 LD (NN),SP
ED78 IN A,(C)
ED79 OUT (C),A
ED7A ADC HL,SP
ED7B LD SP,(NN)
EDA0 LDI
EDA1 CPI
EDA2 INI
EDA3 OUTI
EDA8 LDD
EDA9 CPD
EDAA IND
EDAB OUTD
EDB0 LDIR
EDB1 CPIR
EDB2 INIR
EDB3 OTIR
EDB8 LDDR
EDB9 CPDR
EDBA INDR
EDBB OTDR
FD09 ADD IY,BC
FD19 ADD IY,DE
FD21 LD IY,NN
FD22 LD (NN),IY

FD23	INC IY	FDE9	JP (IY)
FD29	ADD IY,IY	FDF9	LD SP,IY
FD2A	LD IY,(NN)	FDCB 06	RLC (IY+IND)
FD2B	DEC IY	FDCB DE	RRC (IY+IND)
FD34	INC (IY+IND)	FDCB 16	RL (IY+IND)
FD35	DEC (IY+IND)	FDCB 1E	RR (IY+IND)
FD36	LD (IY+IND),N	FDCB 26	SLA (IY+IND)
FD39	ADD IY,SP	FDCB 2E	SRA (IY+IND)
FD46	LD B,(IY+IND)	FDCB 3E	SRL (IY+IND)
FD4E	LD C,(IY+IND)	FDCB 46	BIT 0,(IY+IND)
FD56	LD D,(IY+IND)	FDCB 4E	BIT 1,(IY+IND)
FD5E	LD E,(IY+IND)	FDCB 56	BIT 2,(IY+IND)
FD66	LD H,(IY+IND)	FDCB 5E	BIT 3,(IY+IND)
FD6E	LD L,(IY+IND)	FDCB 66	BIT 4,(IY+IND)
FD70	LD (IY+IND),B	FDCB 6E	BIT 5,(IY+IND)
FD71	LD (IY+IND),C	FDCB 76	BIT 6,(IY+IND)
FD72	LD (IY+IND),D	FDCB 7E	BIT 7,(IY+IND)
FD73	LD (IY+IND),E	FDCB 86	RES 0,(IY+IND)
FD74	LD (IY+IND),H	FDCB 8E	RES 1,(IY+IND)
FD75	LD (IY+IND),L	FDCB 96	RES 2,(IY+IND)
FD77	LD (IY+IND),A	FDCB 9E	RES 3,(IY+IND)
FD7E	LD A,(IY+IND)	FDCB A6	RES 4,(IY+IND)
FD86	ADD A,(IY+IND)	FDCB AE	RES 5,(IY+IND)
FD8E	ADC A,(IY+IND)	FDCB B6	RES 6,(IY+IND)
FD96	SUB (IY+IND)	FDCB BE	RES 7,(IY+IND)
FD9E	SBC A,(IY+IND)	FDCB C6	SET 0,(IY+IND)
FDA6	AND (IY+IND)	FDCB CE	SET 1,(IY+IND)
FDAE	XOR (IY+IND)	FDCB D6	SET 2,(IY+IND)
FDB6	OR (IY+IND)	FDCB DE	SET 3,(IY+IND)
FDBE	CP (IY+IND)	FDCB E6	SET 4,(IY+IND)
FDE1	POP IY	FDCB EE	SET 5,(IY+IND)
FDE3	EX (SP),IY	FDCB F6	SET 6,(IY+IND)
FDE5	PUSH IY	FDCB FE	SET 7,(IY+IND)