contents onto the stack, and the effect of the instructions between SUBR3 and SUBR4 is to restore those contents to the registers. The substantive instructions in the subroutine are the two starting at SUBR2, but the second of these is ineffective since the subsequent instructions completely change the state of the accumulator.

Notice that the Z80 PUSH and POP instructions can take any of the register pairs as an operand, but the 6502 can operate on only the accumulator (PHA and PLA) and the processor status register (PHP and PLP). Hence the need for the register-accumulator transfers (TXA, TAX, TYA, TAY) in the 6502 version. Notice also that we have made a deliberate mistake in the Z80 version in not 'popping' all of the registers in the reverse order to which they were 'pushed'. It illustrates the care needed in stack operations, but also demonstrates that you can push the stack from one register and then pop that value off the stack back into a different register — a laborious but sometimes convenient way of doing data transfers between registers.

The functions and uses of the CPU registers are the subjects of the next instalment, in which we conclude our general examination of the Assembly language instruction set. We also begin the study of machine code arithmetic.

Exercises

1) Rewrite the second routine given in the answers to the previous exercises so that the message at LABL1 is stored back at LABL1, but in reverse order, thus:

LABL1 EGASSEM A SI SIHT

Use the stack for this reversal.

2) Develop this routine so that the words of the message remain in their original order, but the characters of each word are reversed, thus:

LABL1 SIHT SI A EGASSEM

Answers To Exercises On Page 237

1) This subroutine stores the numbers $0F to $00 in descending order in the block of $10 bytes reserved by the DS pseudo-op at LABL1.

6502			Z80		
ORIGIN	ORG	$7000	ORIGIN	ORG	$C000
LABL1	DS	$10	LABL1	DS	$10
LABL2	DW	$7100	LABL2	DW	$C100
			OFFST	EQU	$0F
BEGIN	LDY	#$FF	BEGIN	LD	IX,LABL1
	LDX	#$10		LD	B,OFFST
LOOP0	INY		LOOP0	LD	(IX+0),B
	DEX			INC	IX
	TXA		ENDLP0	DJNZ	LOOP0
	STA	LABL1,Y		LD	(IX+0),B
ENDLP0	BNE	LOOP0		RET	
	RTS				

The differences in approach and instructions between the Z80 and 6502 are revealing. The 6502 uses the Y register as an index to the address LABL1, and the X register as a loop counter and source of the data to be stored. Notice that the X register is decremented two instructions before the BNE test at ENDLP0, but because TXA (Transfer X contents to the Accumulator) and the STA do not affect the processor status register, the test works on the effects of decrementing X.

The Z80 version uses IX indirect addressing mode to hold the storage address, and uses the B register as the counter and source of data. At ENDLP0 in the Z80 version we see DJNZ LOOP0, meaning 'decrement register B, and jump relative to LOOP0 if the result is non-zero'. This instruction is almost an Assembly language FOR...NEXT structure, and certainly makes writing Z80 loops easy and convenient.

2) This routine copies the message stored at LABL1 to the block starting at the address stored at LABL2. The value $0D (the ASCII code for Return or Enter) is stored at the end of the message as a terminator.

6502			Z80		
ORIGIN	ORG	$7000	ORIGIN	ORG	$C000
LABL1	DB	'THIS IS A MESSAGE'	LABL1	DB	'THIS IS A MESSAGE'
TERMN8	DB	$0D	TERMN8	DB	$0D
LABL2	DW	$7100	LABL2	DW	$C100
CR	EQU	$0D	CR	EQU	$0D
ZPLO	EQU	$FB			
BEGIN	LDA	LABL2	BEGIN	LD	IX,LABL1
	STA	ZPLO		LD	IY,(LABL2)
	LDA	LABL2+1	LOOP0	LD	A,(IX+0)
	STA	ZPLO+1		LD	(IY+0),A
	LDY	$FF		INC	IX
LOOP0	INY			INC	IY
	LDA	LABL1,Y		CP	CR
	STA	(ZPLO),Y	ENDLP0	JR	NZ,LOOP0
	CMP	CR		RET	
ENDLP0	BNE	LOOP0			
	RTS				

The 6502 version uses the Y register as an index to the indirect address ZPLO, in the post-indexed indirect mode. This mode is possible only with the Y register, and requires a zero page operand address — hence the initialisation of ZPLO and ZPLO+1 with the address stored at LABL2. The operating system in 6502 machines uses most of the zero page locations, but locations $FB to $FF on the Commodore 64, and $70 to $8F on the BBC Micro, are unused, so ZPLO is set to one of these locations. The Z80 version uses IX in indexed mode, and IY in indexed indirect mode.

Both routines use a 'compare the accumulator' instruction — CMP CR (6502) and CP CR (Z80) — in which the operand is subtracted from the accumulator contents, thus affecting the processor status register (PSR) flags. The accumulator contents are then restored, while the PSR shows the results of the comparison. When the accumulator contains $0D (the message terminator), the result of the comparison will be that the zero flag is set. Thus the ENDLP0 test will fail and control will pass to the return instruction.