**Assembly Programming 51-sample[C]**

more on data manipulation

[1] successive additions of 32-bit data

= = = = = = = = = = = = = = = = = = = =

ORG 0000H

; data1: 01001234H at 30H

; data2: 00105678H at 34H

; data3: 10009ABCH at 38H

; data4: 1001DEF0H at 3CH

; sum: at 40H

; 50H: Cy from additions of lower-

; byte positions

start:

mov R1, #40H

mov R3, #4

mov 50H, #0

mov 51H, #30H

add\_loop1:

push R3 ; or PUSH 3?

clr C

mov R3, #3

mov R0, 51H

mov A, @R0

add A, 50H

push A

mov 50H, #0

clr A

addc A, 50H

mov 50H, A

pop A

add\_loop2:

inc R0

inc R0

inc R0

inc R0

add A, @R0

push A

clr A

addc A, 50H

mov 50H, A

pop A

djnz R3, add\_loop2

mov @R1, A;

inc R1

inc 51H

pop R3

djnz R3, add\_loop1

;jmp start

sjmp $

END

[2] left-shift of a byte with 0-padding at

LSB position.

= = = = = = = = = = = = = = = = = = = =

ORG 0

; left-shift of the byte at 40H by 3

; bits

mov R3, #3

mov A, 40H

pad0\_3:

clr C

rlc A

djnz A, pad0\_3

mov 40H, A

sjmp $

END

[3] left-shift of a 32-bit data entity with 0-padding at LSB position.

= = = = = = = = = = = = = = = = = = = =

ORG 0

; 32-bit data at 30H to be left-shifted

; by 4 bits

mov R0, #30H

mov R3, #4

loop1:

push R3

mov R3, #4

clr C

loop2:

mov A, @R0

rlc A

mov @R0, A

inc R0

djnz R3, loop2

pop R3

djnz R3, loop1

sjmp $

END

[4] right-shift of a 32-bit data with sign-extension at MSB location

= = = = = = = = = = = = = = = = = = = =

ORG 0

; right-shift with sign-extension of a

; 32-bit data at 30H by 4 bits

mov A, 33H

anl A, #80H

rlc A

mov 30H, C

mov R3, #4

loop1:

push R3

mov R0, #33H

mov R3, #4

mov C, 30H

loop2:

mov A, @R0

rrc A

mov @R0, A

dec R0

djnz R3, loop2

pop R3

djnz R3, loop1

sjmp $

END

[5] right-shift of a data with sign-extension at MSB location

= = = = = = = = = = = = = = = = = =

ORG 0

; right-shift with sign-extension of a

; byte data at 30H by 4 bits

mov R0, #30H

mov A, 30H

anl A, #80H

rlc A

mov 30H, C

mov A, @R0

mov R3, #8

loop:

rrc A

mov C, 30H

djnz R3, loop

sjmp $

END

[6] reversal of bit sequence in a data byte at 30H;

\* the byte at 30H remaining intact after the bit-reversing process.

\* result of reversal saved at 31H.

= = = = = = = = = = = = = = = = = = = =

ORG 0

mov 31H, #0

clr C

mov R3, #9

loop:

mov A, 30H

rrc A

mov 30H, A

mov A, 31H

rlc

mov 31H, A

djnz R3, loop

mov A, 31H

rrc

mov 31H, A

sjmp $

END

[7] reversal of bit sequence in a 32-bit data item at 30H;

\* result of reversal saved at 34H.

\* the data at 30H remaining intact after the bit-reversing process.

= = = = = = = = = = = = = = = = = = = =

ORG 0

mov R3, #33

push R3

loop1:

clr C

mov R0, #33H

mov R3, #4

loop2:

mov A, @R0

rrc A

mov @R0, A

dec R0

djnz R3, loop2

jnc cont

orl 33H, #80H

cont:

mov R0, #34H

mov R3, #4

loop3:

mov A, @R0

rlc A

mov @R0, A

inc R0

djnz R3, loop3

pop R3

djnz R3, loop1

mov R3, #4

mov R0, #37H

loop4:

mov A, @R0

rrc A

mov @R0, A

dec R0

djnz R3, loop4

sjmp $

END

[8] multiplication of 16-bit data with

MUL AB

= = = = = = = = = = = = = = = = = = = =

ORG 0

; multiplicand at 30H: 1234H

; multiplier at 32H: 5678H

; product at 40H:

; partialsum1 at 50H: 0000H

; partialsum2 at 54H: 0000H

; partialsum3 at 58H: 0000H

; partialsum4 at 5CH: 0000H

start:

mov 30H, #34H ; M1L

mov 31H, #12H ; M1H

mov 32H, #78H ; M2L

mov 33H, #56H ; M2H

mov R0, #50H

mov R3, #16

init: mov @R0, #0

djnz R3, init

mov A, 30H ; M1L x M2L

mov B, 32H

mul A, B ; or MUL AB ?

mov 51H, B

mov 50H, A ; partialsum1

mov A, 31H ; M1H x M2L

mov B, 32H

mul AB

mov 56H, B

mov 57H, A ; partialsum2

mov A, 30H ; M1L x M2H

mov B, 33H

mul AB

mov 59H, B

mov 5AH, A ; partialsum3

mov A, 31H ; M1H x M2H

mov B, 33H

mul AB

mov 5FH, B

mov 5EH, A ; partialsum4

mov 40H, 50H ; 40H 🡨 sum1

mov 44H, #0

mov A, 51H

add A, 55H

push A

clr A

addc A, 44H

mov 44H, A

pop A

add A, 59H

mov 41H, A ; 41H 🡨 sum2

clr A

addc A, 44H

; mov 44H, A

mov 44H, #0

add A, 56H

push A

clr A

addc A, 44H

mov 44H, A

pop A

add A, 5AH

push A

clr A

addc A, 44H

mov 44H, A

pop A

add A, 5EH

mov 42H, A ; 42H 🡨 sum3

clr A

addc A, 44H

add A, 5FH

mov 43H, A ; 43H 🡨 sum4

sjmp $

END

[9] Write a 51 assembly code piece for the task of 16-bit data multiplication, without using MUL AB instruction.

[10] Write a 51 assembly code piece for the task of 16-bit data division.