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

more on data manipulation

[1] bit-sequence-reversion in every 3-bit fields in a 24-bit data item

b23  b20  b5b4b3 b2b1b0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| F7 | F6 | . . . . | F1 | F0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| F7 | F6 | . . . . | F1 | F0 |

b21  b18  b3b4b5 b0b1b2

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

|  |
| --- |
| b23 . . . b0 |

|  |
| --- |
| b21b22b23 . . . b3b4b5b0b1b2 |

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

ORG 0000H

; input: 24-bit data item at 30H

: output: 24-bit data item at 34H

; temporary working space

; for 3-bit reversion: 40H

;

; OPs: 3-leveled looping

; 8 rounds for outer-loop

; [outer loop]:

; \* 3 rounds of mid1-loop

; [mid1 loop]:

; \* 3 rounds for mid1-inner

; [mid1 inner loop]:

; \* RRC [32H-30H],

; started with 32H,

; 1B-rrc each round

; \* [32H-30H]:LSB into Cy

; \* padding 32H:MSB with Cy

; \* RLC 40H

; \* 3 rounds for mid2-loop

; [mid2 loop]

; \* RRC 40H, 40H:LSB into Cy

; \* 3 rounds for

; [mid2 inner loop]

; \* Cy into [36H]:MSB

; by RRC [36H:34H]

; started with 36H

;

mov 30H, #0BEH

mov 31H, #0C2H

mov 32H, #0CDH

; data initiation done: oct-63341276

; = = = = = = = = = = = = = = = = = =

mov 40H, #0

mov 34H, #0

mov SP, #50H

mov A, 32H

mov R3, #8

outer\_loop: ; \* \* \* \* \* \* \*

push R3

mov R3, #3

mid1\_loop: ; = = = = = = = =

push R3

mov R0, #32H

mov R3, #3

clr C

mid1\_inner\_loop: ; - - - - - - -

mov A, @R0

rrc A

mov @R0, A

dec R0

djnz R3, mid1\_inner\_loop

;- - - - - - - -

jnc nc3

orl 32H, #80H

nc3:

mov A, 40H

rlc A

mov 40H, A

pop R3

djnz R3, mid1\_loop

; = = = = = = =

mov R3, #3

mid2\_loop: ;; = = = = = = =

push R3

mov A, 40H

rrc A

mov 40H, A

mov R0, #36H

mov R3, #3

mid2\_inner\_loop: ;; - - - - - - - -

mov A, @R0

rrc A

mov @R0, A

dec R0

djnz R3, mid2\_inner\_loop

;; - - - - - - - -

pop R3

djnz R3, mid2\_loop

;; = = = = = =

pop R3

djnz R3, outer\_loop

; \* \* \* \* \* \* \*

sjmp $

[2] field-sequence-reversion of every 3-bit fields in a 24-bit data item

b23  b20  b5b4b3 b2b1b0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| F7 | F6 | . . . . | F1 | F0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| F0 | F1 | . . . . | F6 | F7 |

b2b1b0  b5b4b3 b20 b23

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

ORG 0

; input: 3-byte data item starting at 30H

; (24-bit equivalent), organized

; as 8 fields of 3-bit data.

; output: 3-byte data item starting at

; 34H (24-bit equivalent)

; temp-buffer: 40H for one 3-bit field

mov SP, #50H

mov R3, #8

loop1:

push R3

mov 40H, #0

mov R3, #3

loop21:

push R3

mov R3, #3

mov R0, #32H

clr C

loop321:

mov A, @R0

rrc A

mov @R0, A

dec R0

djnz R3, loop321

jnc nc321

orl 32H, #80H

nc321:

mov A, 40H

rrc A

mov 40H, A

pop R3

djnz loop21, R3

; 3-bit in 40H:[b7~b5] done

; = = = = = = = = = = = = = = = = =

mov R3, #3

clr C

loop22:

push R3

mov A, 40H

rlc A

mov 40H, A

mov R3, #3

mov R0, #34H

loop322:

mov A, @R0

rlc A

mov @R0, A

inc R0

djnz R3, loop322

pop R3

djnz R3, loop22

; Fj+1 🡨 Fj ; done

; F0 🡨 40H:[b7~b5] done

; = = = = = = == = = = = = = = = =

pop R3

djnz R3, loop1

pop R3

djnz R3, loop1

; RRC(input”)🡪 BUF(40H)

; RLC(output)🡨BUF(40H)

; ops for the next 3-bit field

; = = = = = = = = = = = = = = =

sjmp $

END

[3] reversing byte-order in a data block of 16 data bytes, starting at 30H.

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

ORG 0

; input: 16-byte data block at 30H

; output: the same space as of the input

mov SP, #50H

mov R0, #30H

mov R3, #16

loop1:

mov A, @R0

push A

inc R0

djnz R3, loop1

mov R0, #30H

mov R3, #16

loop2:

mov A, @R0

pop A

inc R0

djnz R3, loop2

sjmp $

[4] Reversing data-order in a data structure within a scope without altering data-order at levels higher above..

Consider a data block, staring at 30H, of 15-byte long organized as 5 3-byte banks, as shown below. Reversing orders of data items below the bank-level as shown below

Original byte-sequence in the block: A-B-C- . . .-N-O.

3EH 30H

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| O | N | M | L | K | J | I | H | G | F | E | D | C | B | A |

BNK4 BNK3 BNK2 BNK1 BNK0

After the reversing process:

3EH 30H

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M | N | O | J | K | L | G | H | I | D | E | F | A | B | C |

BNK4 BNK3 BNK2 BNK1 BNK0

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

**[prototype]**

ORG 0

; input: data block of 15 bytes, starting

; at 30H, organized as 5 banks,

; each being 3-byte in length

; output: data block of 15 bytes,

; occupying the same space as

; the input, with byte-order in

; each bank reversed.

mov SP, #50H

mov R0, #30H

mov R3, #15

loop1:

mov A, @R0

push A

inc R0

djnz R3, loop1

; data block saved in the STACK

; = = = = = = = = = = = = = = = = = = mov R0, #3EH

mov R3, #5

loop2:

mov 40H, R3 ; push R3?

mov R3, #3

dec R0

dec R0

loop3: ; - - - - - - - - - - -

pop A

mov @R0, A

inc R0

djnz R3, loop3

; 3-byte in a bank

; reversed and in

; position

; - - - - - - - - - - -

mov R3, 40H ; pop R3?

dec R0

dec R0

dec R0

dec R0

djnz R3, loop2

sjmp $

END

**[alternative scheme]**

ORG 0

mov SP, #50H

mov R0, #30H

mov R1, #30H

mov R3, #5

loop1:

push R3

mov R3, #3

loop2:

mov A, @R0

push A

inc R0

djnz R3, loop2 ; 3 PUSH’s

mov R3, #3

loop3:

pop A

mov @R1, A

inc R1

djnz R3, loop3 ; 3 POP’s

pop R3

djnz r3, loop1

sjmp $

; the alternative, as compared to the

; one in [prototype], possesses vantages

; below:

; code structure: neater

; index control: far less complicated

; STACK: less space

; data operations: far more

; comprehensible

[observations]

Problem solving schemes conceived

differently may lead to great deviations

among the code implementations.

[5] reversing data-order in a data structure within a scope without altering data-order at levels lower below.

Consider a data block, staring at 30H, of 15-byte long organized as 5 3-byte banks, as shown below. Reversing orders of data items below the bank-level as shown below

Original byte-sequence in the block: A-B-C- . . .-N-O.

3EH 30H

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| O | N | M | L | K | J | I | H | G | F | E | D | C | B | A |

BNK4 BNK3 BNK2 BNK1 BNK0

After the reversing process:

3EH 30H

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | B | A | F | E | D | I | H | G | L | K | J | O | N | M |

BNK0 BNK1 BNK2 BNK3 BNK4

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

; input: data block of 15 bytes, starting

; at 30H, organized as 5 banks,

; each being 3-byte in length

; output: data block of 15 bytes,

; occupying the same space as

; the input, with bank-order

; reversed while byte-order

; within each bank unchanged.

**[prototype]**

ORG 0

mov SP, #50H

mov R0, #30H

mov R3, #15

loop1:

mov A, @R0

push A

inc R0

djnz R3, loop1

; 15-byte data block saved in the STACK

; = = = = = = = = = = = = = = = = = = = = = =

mov R0, #30H

mov R3, #5

loop2:

mov 40H, R3 ; push R3?

inc R0

inc R0

mov R3, #3

loop3: ; - - - - - - - - - - - - - - - - -

pop A

mov @R0, A

dec R0

djnz R3, loop3

; - - - - - - - - - - - - - - - - - -

; 3-byte bank in place

mov R3, 40H ; pop R3?

inc R0

inc R0

inc R0

inc R0

djnz R3, loop2

sjmp $

END

[2nd thought]

1. what if the operations in LOOP1 pushing the 15-byte data block onto the STACK in the revered order:

O-N-M-L-K-J-I-H-G-F-E-D-C-B-A?

1. plausible alternatives?

**[alternative scheme]**

ORG 0

mov SP, #50H

mov R0, #3EH

mov R1, #30H

mov R3, #5

loop:

mov 40H, R3

mov R3, #3

loop1:

mov A, @R0

push A

inc R0

djnz R3, loop1

mov R3, #3

loop2:

pop A

mov @R1, A

inc R1

djnz R3, loop2

mov R3, 40H

djnz R3, loop

sjmp $

[observations]

???