

ASSEMBLY REVISION 3 ANSWER BY Abdelrahman ghoniem:

Question 1

Repeated in revision 2

Discuss the types of Registers of the Processor 8086.

1. **Data Registers (AX, BX, CX, DX):** Used for general-purpose data manipulation.
2. **Pointer Registers (SI, DI, BP, SP):** Used for offset storage and addressing in certain instructions.
3. **Index Registers (SI, DI):** Used for indexed addressing, especially in string manipulation instructions.
4. **Segment Registers (CS, DS, SS, ES):**

Hold segment addresses for different types of memory access.

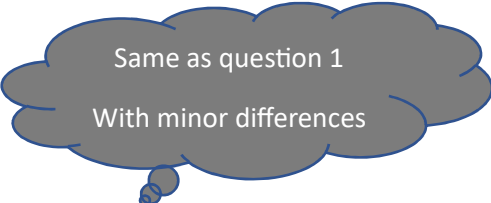
CS (Code Segment) points to the segment containing the program code,

DS (Data Segment) to the segment containing data,

SS (Stack Segment) to the segment containing the stack,

ES (Extra Segment) is an additional segment register.

5. **Flag Register (FLAGS):** Contains status flags such as Zero Flag (ZF), Sign Flag (SF), Overflow Flag (OF), Carry Flag (CF), Parity Flag (PF), and others.
6. **Instruction Pointer (IP):** Points to the next instruction to be executed within the current code segment.
7. **Stack Pointer (SP):** Points to the top of the stack, used for stack operations.



Same as question 1

With minor differences

Question 2

Discuss the Assembly Code Structure & which Registers are used with each Segment.

Assembly code typically follows a structure that includes the following sections:

- **Data Segment (DATA):** This segment is used to define data variables. The DS (Data Segment) register is used to point to this segment.
- **Code Segment (CODE):** This segment contains the actual assembly instructions. The CS (Code Segment) register points to this segment.
- **Stack Segment (STACK):** This segment is used for stack operations. The SS (Stack Segment) register points to this segment.
- **Extra Segment (EXTRA):** This segment is an additional segment used for certain operations. The ES (Extra Segment) register points to this segment.

Registers used with each segment:

- **DS (Data Segment):** Used to access data in the Data Segment.
- **CS (Code Segment):** Used to access instructions in the Code Segment.
- **SS (Stack Segment):** Used for stack operations.
- **ES (Extra Segment):** Used for specific operations that require an additional segment.

Question 3

Calculate the address of the next instruction to be executed for the following: CS = 00D9H & IP = 0036H.

Next Instruction Address=CS+IP

Next Instruction Address=00D9H+0036H

Next Instruction Address=010FH

Question 4

What is the address determined by the sum of SS and SP registers?

Stack Address=SS×16+SP Stack Address=SS+SP

This gives the address of the top of the stack.

Question 5

What is the initial value in SP register? If SP contains 40H and AX contains 43ABH, & BX = 2B5Ah, determine the contents of the stack and value of SP after execution of the instruction: PUSH AX & PU BX, POP BX, & POP AX.

The initial value in the SP register is determined by the operating system or the program loader during the program's startup. It points to the top of the stack.

1. PUSH AX:

- Contents of AX (43ABH) are pushed onto the stack.
- SP is decremented by 2.

2. PUSH BX:

- Contents of BX (2B5AH) are pushed onto the stack.
- SP is decremented by 2.

3. POP BX:

- Contents of the top of the stack are popped into BX.
- SP is incremented by 2.

4. POP AX:

- Contents of the new top of the stack are popped into AX.
- SP is incremented by 2.

Doctor's SOL.

START:-

| | | |
|----|----|--------|
| - | - | 42 |
| 00 | 00 | 40<-SP |
| 00 | 00 | 38 |
| 00 | 00 | 36 |
| 00 | 00 | |

2-PUSH BX

| | | |
|----|----|-----------|
| - | - | 42 |
| AB | 43 | 40 |
| 5A | 2B | 38 |
| 00 | 00 | 36<-SP=36 |
| 00 | 00 | BX=2B5A |

4-POP BX

| | |
|----|----|
| - | - |
| AB | 43 |
| 5A | 2B |
| 00 | 00 |
| 00 | 00 |

1-PUSH AX

| | |
|----|----|
| - | - |
| AB | 43 |
| 00 | 00 |
| 00 | 00 |
| 00 | 00 |

3-POP BX

| | |
|----|----|
| - | - |
| AB | 43 |
| 5A | 2B |
| 00 | 00 |
| 00 | 00 |

| |
|--------|
| 42 |
| 40 |
| 38->SP |
| 36 |

AX=43AB

| |
|--------------|
| 42 |
| 40 |
| 38<-SP |
| 36 <u>+2</u> |

BX=2B5A

| |
|---------|
| 42 |
| 40<-SP |
| 38 |
| 36 |
| AX=42AB |

Question 6

Write the Assembly Code to solve the equation: $Y = A + B - 4 * C / D$, where $A = 25$, $B = 37$, $C = 7$, and $D = 2$.

Using the Conventional Segment Directive:

Page 60,130

Title Equation Solver

S1 segment PARA STACK 'Stack'

Dw 32 DUP(0)

S1 ends

D1 segment PARA 'DATA'

A DB 25

B DB 37

C DB 7

D DB 2

Y DB ?

D1 ends

C1 segment PARA 'Code'

Main proc FAR

ASSUME SS:S1, DS:D1, CS:C1

MOV AX, D1

MOV DS, AX; Equation: $Y = A + B - 4 * (C / D)$

MOV AL, A

ADD AL, B

MOV AH, 0

MOV BL, 4

MUL BL ; $AX = AX * BL (4 * (A + B))$

MOV BL, C

MOV BH, 0

DIV D ; $AX = AX / D (4 * (C / D))$

SUB AL, AH ; Subtract the result from the previous sum

MOV Y, AL ; Store the result in Y

INT 21H ; You may want to use INT 21H for DOS services

MOV AX, 4C00H

INT 21H

Main endp

C1 ends

End Main

Using the Simplified Segment Directive:

```
.MODEL SMALL
```

```
.STACK 64
```

```
.DATA
```

```
    A DB 25
```

```
    B DB 37
```

```
    C DB 7
```

```
    D DB 2
```

```
    Y DB ?
```


.CODE

Main PROC FAR

MOV AL, A ; AL = A

ADD AL, B ; AL = A + B

MOV AH, 0

MOV BL, 4

MUL BL ; AX = AX * BL (4 * (A + B))

MOV BL, C

MOV BH, 0

DIV D ; AX = AX / D (4 * (C / D))

SUB AL, AH ; Subtract the result from the previous sum

MOV Y, AL ; Store the result in Y

INT 21H ; You may want to use INT 21H for DOS services

MOV AX, 4C00H

INT 21H

Main ENDP

END Main

Question 7

Write an assembly code to add the N integer numbers from 1 to N, where N = 95, using the Zero Flag in the iteration Loop for N times. Calculate the sum and average.

Using the Conventional Segment Directive:

Page 60,130

TITLE average

S1 segment PARA STACK 'Stack'

Dw 32 DUP(0)

S1 ends

D1 segment PARA 'DATA'

Ary DB 1,2,3,4,...,95

Sum DB 0

Avg DB 0

D1 ends

Explanation:

PARA STACK-> define that this is stack segment

Dw 32 DUP(0)-> DW stands for "Define Word." define an array of 32 words (16-bit values), with each word initialized to 0.

PARA DATA-> define that this is data segment

C1 segment PARA 'Code'

Main proc FAR

ASSUME SS:S1, DS:D1, CS:C1

MOV AX, D1

MOV DS, AX

MOV SI, OFFSET Ary

MOV CX, 95

MOV Sum, 0 ; Initialize sum to zero

SumLoop:

ADD Sum, [SI] ; Add the current value to sum

INC SI

DEC CX

JNZ SumLoop ; Jump if CX is not zero

MOV AL, Sum

MOV AH, 0

MOV BL, 95

DIV BL ; $AX = AX / BL$ (average)

MOV Avg, AL

INT 21H ; You may want to use INT 21H for DOS services

MOV AX, 4C00H

INT 21H

Main endp

C1 ends

End Main

Explanation:

-Main proc FAR : is like int main()

-ASSUME->tells the assembler the purpose of each segment in the proc

-Segment:special area in the program that contains stack,data,code

-Proc is like (program)

-MOV AX,D1-> moves the D1 value to AX register

-SumLoop: Marks the beginning of a loop.

-ADD Sum, [SI]: Adds the value at the memory location pointed to by SI to the Sum variable.

-INC SI: Increments the source index (SI) to point to the next element in the array.

-DEC CX: Decrements the count register (CX).

-JNZ SumLoop: Jumps back to the SumLoop label if CX is not zero, which means it will repeat the addition until CX becomes zero.

-DIV BL: Divides AX by BL, and the result is stored in AL. This effectively calculates the average of the elements.

-MOV AX, 4C00H: Loads the exit code for the program into AX.

-INT 21H: Calls another DOS interrupt to terminate the program.

Simplified segment Directive:

Page 60 , 132

Titel: Add N integer Numbers

.MODEL SMALL

.STACK 64

.DATA

Ary DB 1,2,3,4,...,95

Sum DB 0

Avg DB 0

.CODE

Main PROC FAR

MOV SI, OFFSET Ary

MOV CX, 95

MOV Sum, 0 ; Initialize sum to zero

SumLoop:

ADD Sum, [SI] ; Add the current value to sum

INC SI

DEC CX

JNZ SumLoop ; Jump if CX is not zero

MOV AL, Sum

MOV AH, 0

MOV BL, 95

DIV BL ; $AX = AX / BL$ (average)

MOV Avg, AL

INT 21H ; You may want to use INT 21H for DOS services

MOV AX, 4C00H

INT 21H

Main ENDP

END Main