20DCS103

Practical : 1

Date:

Aim : Introduction to 8086 micro-processor and assembly language programming.

Ans :

→ The architecture of 8086 is divided into 2 parts.
1) BIU : Bus Interface Unit
2) EU : Execution Unit

1] BIU : Bus Interface Unit :

→ BIU provides interface of 8086 to other devices.

* Operations performed by BIU :

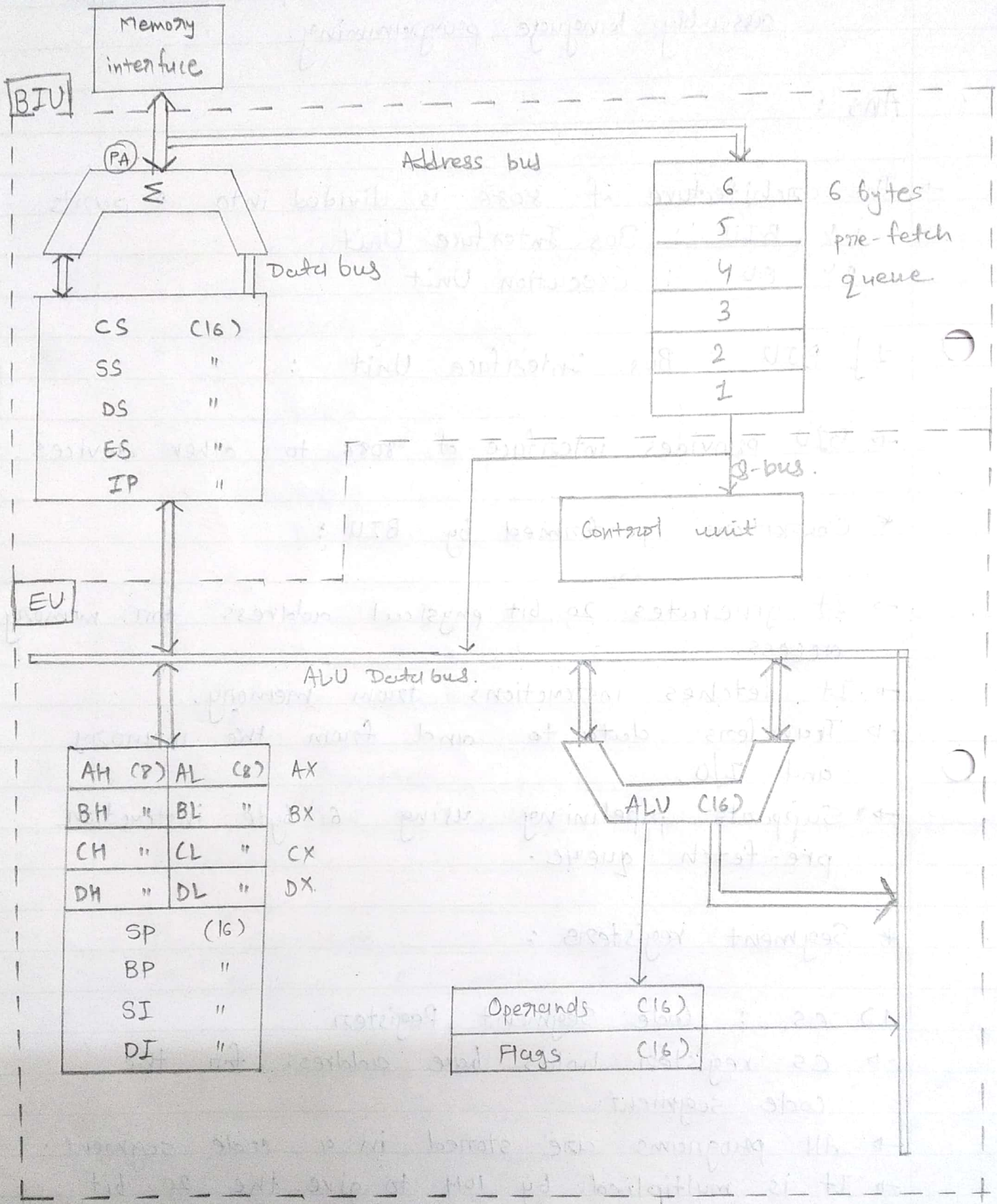
→ It generates 20 bit physical address for memory access.
→ It fetches instructions from memory.
→ Transfers data to and from the memory and I/O.
→ Supports pipelining using 6-byte instruction pre-fetch queue.

* Segment registers :

1) CS : Code Segment Register
→ CS register holds base address for the code segment.
→ All programs are stored in a code segment.
→ It is multiplied by 10H to give the 20 bit physical address to the code segment.

YASH

Royal ECO



Date:

2) DS : Data segment register.
→ DS register holds base address for the data segment.
→ It is multiplied by 10H to provide 20 bit physical address to the data segment.

3) SS : Stack Segment Register
→ SS register holds base address for stack segment.
→ It is multiplied by 10H to give 20 bit physical address to the stack segment.

4) ES : Extra Segment register.
→ ES register holds base address for extra segment.
→ It is multiplied by 10H to give 20 bit physical address to the extra segment.

* Instruction Pointer : IP : (16 bit)

→ IP holds offset of the next instruction in the code segment.
→ IP is incremented after every instruction byte is fetched.
→ It gets a new value whenever a branch occurs.

* 6-byte pre-fetch queue :

→ Fetching the next instruction while executing the current instruction is called pipelining.

YASH

Royal ECO

Date:

→ BIU fetches the next 6-byte instruction from code segment and stores it into the queue. EU removes instructions from the queue and executes it.

→ The queue will be refilled when atleast 2 bytes are empty as 8086 has 16 bit data bus.

2] EU : Execution Unit :

→ EU fetches the instructions from queue in BIU and decodes and executes it.

→ It performs logical, arithmetic and internal data transfer operations.

→ It operates w.r.t. T-states clock cycle.

* General purpose registers :

→ AX register : Accumulator Register.
  → holds operands and results during multiplication and division.
  → functions as accumulator during string operations

→ BX register : Base Register
  → holds offset address in indirect addressing mode.

Date:

→ CX register : Count Register
→ Used as counter for the operations like loop, rotate, shift, etc...

→ DX register : Data Register
→ used with AX to hold 32 bit values during multiplication and division.

* Special purpose register :

→ SP : Stack Pointer
→ BP : Base Pointer
→ SI : Source Index
→ DI : Destination Index

* ALU : Arithmetic Logic Unit (16 bit) :
→ It performs 8 bit and 16 bit arithmetic and logic operations.

* Operand Register :
→ It is used by control register to hold operands temporarily.

* Flag Register :
→ 9 flags : 6 - status flags
3 - control flags
→ Flags give the status of the current result

Practical : 2

Date:

Aim : Store the data byte 32 H into memory location 4000 H

Assembly code :

```
org 100h
MOV AX, 00
MOV DS, AX
MOV AX, 32h
MOV 4000h, AX
ret
```

YASH

Royal ECO

* Data registers :

	H	L
AX	00	32
BX	00	00
CX	00	00
DX	00	00

* Base address : offset = 00 : 4000

* physical memory location : data

=

04000 :	32	050

Practical : 3

Date:

Aim : Exchange the contents of memory locations 2000 H and 4000 H.

Assembly code :

```
org  100h
MOV  AX , 00
MOV  DS , AX
MOV  AX , 2
MOV  BX , 5
MOV  2000H , AX
MOV  4000H , BX
XCHG  AX , BX
MOV  2000H , AX
MOV  4000H , BX
ret
```

YASH

Royal ECO

* Data registers :

	H	L
AX	00	05
BX	00	02
CX	00	1D
DX	00	00

* Base address : offset = 00 : 2000

02000 :	05	005

* Base address : offset = 00 : 4000

04000 :	02	002

Practical : 4

Date:

Aim : Convert the below given C program into Assembly Language.

```
main()
{
    int l, m, n, o, p;
    l = m + n - o + p;
}
```

Assembly code :

```
org 100h
MOV AX, 00
Mov DS, AX
MOV AL, 00
MOV BL, 10
MOV CL, 8
MOV DL, 9
MOV AH, 7
ADD BL, CL
SUB BL, DL
ADD BL, AH
MOV AL, BL
MOV 4000H, AL
ret
```

* Data registers :

	H	L
AX	07	10
BX	00	10
CX	00	08
DX	00	09

* Base address : offset = 00 : 4000

*. Physical memory location : data

04000 :	10	016

Practical : 5

Date:

Aim : Subtract the contents of memory location 4001 H from the memory location 2000 H and place the result in memory location 4002 H.

Assembly code :

```
org 100h
MOV AX, 00
MOV DS, AX
MOV AL, 18h
MOV BL, 12h
MOV 2000H, AL
MOV 4001H, BL
SUB AL, BL
MOV 4002H, AL
ret
```

* Data registers :

	H	L
AX	00	06
BX	00	12
CX	00	18
DX	00	00

* Base address : offset = 00 : 4002.

* Physical memory location : data

04002 :	06	006.

Practical : 6

Date:

Aim : Add the 16 bit number in memory locations 4000H and 4001H to the 16 bit number in memory locations 4002H and 4003H. The most significant eight bits of the two numbers to be added are in memory locations 4001H and 4003H. Store the result in memory locations 4004H and 4005H with the most significant byte in memory location 4005H.

```
org  100h
MOV  AX, 00
MOV  DS, AX
MOV  AH, 10h
MOV  AL, 20h
MOV  BH, 30h
MOV  BL, 40h

MOV  4000H, AL
MOV  4001H, AH
MOV  4002H, BL
MOV  4003H, BH

ADD  AL, BL
ADDC BAH, BH
MOV  400HH, AL
MOV  4005H, AH
ret
```

* Data registers :

	H	L
AX	40	60
BX	30	40
CX	00	2A
DX	00	00

* Base address : offset = 00 : 4004

* Physical memory location : data

04004 :	60	096

Practical : 7

Date:

Aim : Add the 16 bit number in memory locations 4000H and 4001H to the 16 bit number in memory locations 4002 H and 4003H. The most significant eight bits of the two numbers to be subtracted are in memory locations 4001 H and 4003 H. Store the result in memory locations 4004 H and 4005 H with the most significant byte in memory location 4005 H.

```
org   100h
mov   AX, 00
mov   DS, AX
Mov   AH, 10h
mov   AL, 20h
mov   BH, 30h
mov   BL, 40h

mov   4000H, AL
mov   4001H, AH
mov   4002H, BL
mov   4003H, BH

SUB   AL, BL
SBB   AH, BH
mov   4004H, AL
mov   4005H, AH
ret
```

YASH

Royal ECO

* Data registers :

	H	L
AX	DF	E0
BX	30	40
CX	00	2A
DX	00	00

* Base address : offset = 00 : 4004

* Physical memory location : data

04004 :	E0	224.

Practical : 8

Date:

Aim : Add two 32 bit numbers stored in consecutive memory locations and store the result in memory locations starting from 7000 H.

Assembly code :

```
org 100h
MOV AX, 00
MOV DS, AX

MOV AL, 10h
MOV BL, 20h
MOV CL, 30h
MOV DL, 40h
MOV AH, 50h
MOV BH, 60h
MOV CH, 70h
MOV DH, 80h

MOV 7000H, AL
MOV 7001H, BL
MOV 7002H, CL
MOV 7003H, DL
MOV 7004H, AH
MOV 7005H, BH
MOV 7006H, CH
MOV 7007H, DH
ADD DH, DL
ADD CH, CL
ADD BH, BL
ADD AH, AL
MOV 7008H, AH
MOV 7009H, BH
MOV 700AH, CH
MOV 700BH, DH
ret
```

YASH

Royal ECO

* Data registers :

	H	L
AX	60	10
BX	80	20
CX	A0	30
DX	C0	40

* Base address : offset

00 : 7008 →
00 : 7009 →
00 : 700A →
00 : 700B →

* Physical memory : data location

07008 :	60	096
07009 :	80	128
0700A :	A0	160
0700B :	C0	192

Practical : 9

Date:

Aim : Subtract two 32 bit numbers stored in consecutive memory locations and store the result in memory locations starting from 7000H.

Assembly code :

```
org 100h
MOV AX, 00
MOV DS, AX

MOV AL, 10h
MOV BL, 20h
MOV CL, 30h
MOV DL, 40h
MOV AH, 50h
MOV BH, 60h
MOV CH, 70h
MOV DH, 80h

MOV 7000H, AL
MOV 7001H, BL
MOV 7002H, CL
MOV 7003H, DL
MOV 7004H, AH
MOV 7005H, BH
MOV 7006H, CH
MOV 7007H, DH

SUB DH, DL
SBB CH, CL
SBB BH, BL
SBB AH, AL

MOV 7008H, AH
MOV 7009H, BH
MOV 700AH, CH
MOV 700BH, DH
ret
```

* Data registers :

	H	L
AX	40	10
BX	40	20
CX	40	30
DX	40	40

* Base address : offset

00 : 7008
00 : 7009
00 : 700A
00 : 700B

* Physical memory address : data

07008 :	40	064
07009 :	40	064
0700A :	40	064
0700B :	40	064

Practical : 10

Date:

Aim : Write an assembly language program to convert temperature in F to C.
$C = (F - 32) * 5/9$.

Assembly code :

```
org 100h

MOV AL, 113
SUB AL, 32
MOV BL, 5
IMUL BL
MOV CL, 9
IDIV CL

ret
```

```
org 100h

MOV AX, 00H
MOV DS, AX

MOV AL, 113
SUB AL, 32
MOV BL, 5
IMUL BL
MOV CL, 9
IDIV CL
MOV 4000H, AL
ret
```

* Data registers :

	H	L
AX	00	2D
BX	00	05
CX	00	09
DX	00	00

*. Base address : offset = 00 : 4000

* Physical memory location : data

04000 :	2D	045

* Extended value viewer :

unsigned : 45
signed : 45
ascii : -

Practical : 11

Date:

Aim : Write a program to perform selective set operation on data stored at 4000 H with the data stored at 4001H and store the result at 4002H. Verify the result and write bite wise operation of this program. (OR)

Assembly code :

```
org 100h
MOV AX, 00
MOV DS, AX

MOV AL, 11001010b
MOV BL, 10101011b

MOV 4000H, AL
MOV 4001H, BL

OR AL, BL
MOV 4002H, AL

ret
```

* Data registers :

	H	L
AX	00	EB
BX	00	AB
CX	00	18
DX	00	00

* Base address : offset = 00 : 4002.

* Physical memory address : data

04002 :	EB	235

* Extended value viewer :

	L
hex :	EB
bin :	1110 1011
oct :	353

decimal 8 bit :

unsigned :	235
signed :	-21
ascii :	δ

Practical : 12

Date:

Aim : Write a program to perform selective compliment operation on data stored at 4000H corresponding to the data stored at 4001H and store the result at 4002H. Verify the result and write bit wise operation of this program (XOR).

Assembly code :

```
org 100h
MOV AX, 00
MOV DS, AX

MOV AL, 11001010b
MOV BL, 10101011b

MOV 4000H , AL
MOV 4001H , BL

XOR AL, BL
MOV 4002H , AL
ret
```

* Data register :

	H	L
AX	00	61
BX	00	AB
CX	00	18
DX	00	00

* Base address : offset = 00 : 4002

* Physical memory address : data

04002 :	61	097

* Extended value viewer :

	L
hex :	61
bin :	01100001
oct :	141

decimal 8 bit

unsigned :	97
signed :	97
ascii :	a

Practical : 13

Date:

Aim : Write a program to perform selective clear operation on data stored at 4000H corresponding to the data stored at 4001H and store the result at 4002H. Verify the result and write bit wise operation of this program (A AND B').

Assembly code :

```
org 100h
MOV AX, 00
MOV DS, AX

MOV AL, 11001011b
MOV BL, 10101011b

MOV 4000H, AL
MOV 4001H, BL

NOT BL
AND AL, BL
MOV 4002H, AL
ret
```

YASH

Royal ECO

* Data registers :

	H	L
AX	00	40
BX	00	54
CX	00	1A
DX	00	00

*. Base address : offset = 00 : 4002

* Physical memory address : data

04002 :	40	064

* Extended value viewer :

	L
hex :	40
bin :	01000000
oct :	100

decimal & bit :

unsigned :	64
signed :	64
ascii :	@

Practical : 14

Date:

Aim : Write an assembly language program and manipulate the data stored at memory locations 2000H and 2001H. (Use XOR)

Assembly code :

```
org 100h

MOV AX, 00H
MOV DS, AX

MOV [2000H], 1010B
MOV [2001H], 1100B

MOV AL, [2000H]
MOV BL, [2001H]

XOR AL, BL
MOV [2002H], AL

ret
```

* Data registers :

	H	L
AX	00	06
BX	00	0C
CX	00	1C
DX	00	00

* Base address : offset = 00 : 2002

* Physical memory location : data

02002 :	06	006.

* Extended value viewer :

hex :	06
bin :	0000 0110
oct :	006

decimal & bit :

unsigned :	6
signed :	6
ascii :	♠

Practical : 15

Date:

Aim : Write a program to multiply and devide the number stored at 4000H by 2 and store the result at 4001H and 4002H. (Use the shift instructions)

Assembly code :

```
org 100h
MOV AX, 00
MOV DS, AX

MOV AL, 01001000b
MOV 4000H, AL
SHL AL, 1
MOV 4001H, AL
MOV AH, [4000H]
SHR AH, 1
MOV 4002H, AH
ret
```

YASH

Royal ECO

* Data registers :

	H	L
AX	24	90
BX	00	00
CX	00	1C
DX	00	00

* Base address : offset = 00 : 4002

* Physical memory location : data

= | 04002 : | 24 | 036 |

Practical : 16

Date:

Aim : Write a program to subtract the contents of memory location 4001H from the memory location 4002H and place the result in memory location 4003H without SUB instruction.

Assembly code :

```
org 100h
MOV AX, 00H
MOV DS, AX
MOV AL, 20H
MOV [4001H], AL
MOV BL, 10H
MOV [4002H], BL
NEG BL
ADD AL, BL
MOV [4003H], AL
ret
```

* Data registers :

	H	L
AX	00	10
BX	00	F0
CX	00	18
DX	00	00

* Base address : offset = 00 : 4003

* Physical memory location : data

=

04003 :	10	16

Practical : 17

Date:

Aim : Implement a program to mask the lower four bits of content of the memory location.

Assembly code :

```
org 100h
MOV AX, 00
MOV DS, AX
MOV 4002H, 17H
MOV AL, [4002H]
AND AL, 0F0H
MOV 4003H, AL
ret
```

YASH

Royal ECO

* Data registers :

	H	L
AX	00	10
BX	00	00
CX	00	14
DX	00	00

* Base address : offset = 00 : 4003

* Physical memory location : data.

= | 04003 : | 10 | 016 |

* Extended value viewer :

hex :	10
bin :	0001 0000
oct :	020

decimal 8 bit :

unsigned :	16
signed :	16
ascii :	►

Practical : 18

Date:

Aim: Implement a program to set higher four bits of content of the memory location to 1.

Assembly code :

```
org 100h
MOV AX, 00
MOV DS, AX
MOV 4002H, 17H
MOV AL, [4002H]

OR AL, 0F0H
MOV 4003H, AL
ret
```

* Data registers :

	H	L
AX	00	F7
BX	00	00
CX	00	14
DX	00	00

* Base address : offset = 00 : 4003.

* Physical memory location : data

= | 04003 : | F7 | 247 |

* Extended value viewer :

hex :	F7
bin :	11110111
oct :	367

decimal 8 bit :

unsigned:	247
signed :	-9
ascii :	≈

Practical : 19

Date:

Aim : Calculate the sum of series of numbers (Data set - 1) from the memory location listed below and store the result at 400AH location.

Assembly code :

```
org 100h
MOV AX, 00
MOV DS, AX

MOV [2000H], 11
MOV [2001H], 08
MOV [2002H], 13
MOV [2003H], 14
MOV [2004H], 18

MOV SI, 2000H
MOV CL, 5
MOV AL, [SI]

L1:
    INC SI
    MOV BL, [SI]
    ADD AL, BL
    LOOP L1

MOV 400AH, AL

ret
```

YASH

Royal ECO

* Data registers :

	H	L
AX	00	40
BX	00	00
CX	00	00
DX	00	00

* Base address : offset = 00 : 400AH

* Physical memory address : data

0400A :	40	064

Practical : 20

Date:

Aim : Modify the 19's program in such a way that it halts the execution if carry generated and stores the intermediate result at 400AH location. (Data set - 2)

[Note : student need to implement FOR loop in this program : initialization, Compare, Increment/ Decrement ; also need to use JMP, JMx instructions.]

Assembly code :

```
org 100h

MOV AX, 00H
MOV DS, AX

MOV [2000H], 11
MOV [2001H], 08
MOV [2002H], 13
MOV [2003H], 14
MOV [2004H], 18

MOV SI, 2000H
MOV DL, 5
MOV AL, [SI]

L1 :
   INC SI
   MOV BL, [SI]
   ADD AL, BL
   JC L2
   DEC DL
   JNZ L1

L2 :
   MOV 400AH, AL

ret
```

YASH

Royal ECO

* Data registers :

	H	L
AX	00	40
BX	00	00
CX	00	35
DX	00	00

* Base address : offset = 00 : 400A

* Physical memory location : data

= | 0400A : | 40 | 064 |