

CPU

- **The CPU**

- 4 General Purpose Registers called AX, BX, CX and DX
 - Used for holding values that are going to be manipulated
- Arithmetic Logic Unit (ALU)
 - Executes the arithmetic or logical operation as specified in the instruction
- Instruction Register (IR)
 - Holds the next instruction to be executed
- Instruction Pointer (PC)
 - Holds the address value (location) of the next instruction
- Flags Register
 - Holds information on the outcome of internal operations

Structure of PC subsystems

- **Memory**

- 8 bits wide
- address means location
“where is stored”
- data means
“what is stored”
- One instruction (data)
can take up to three
address locations
- Instructions are copied
one at a time to the CPU
where they are executed

Address Number

MEMORY CHIP

0000
0001
0002
0003
0004
0005
0006
0007
0008

[mov	ax]
[01]
[00]
[mov	bx]
[01]
[00]
[add	ax]
[bx]
[<i>empty</i>]

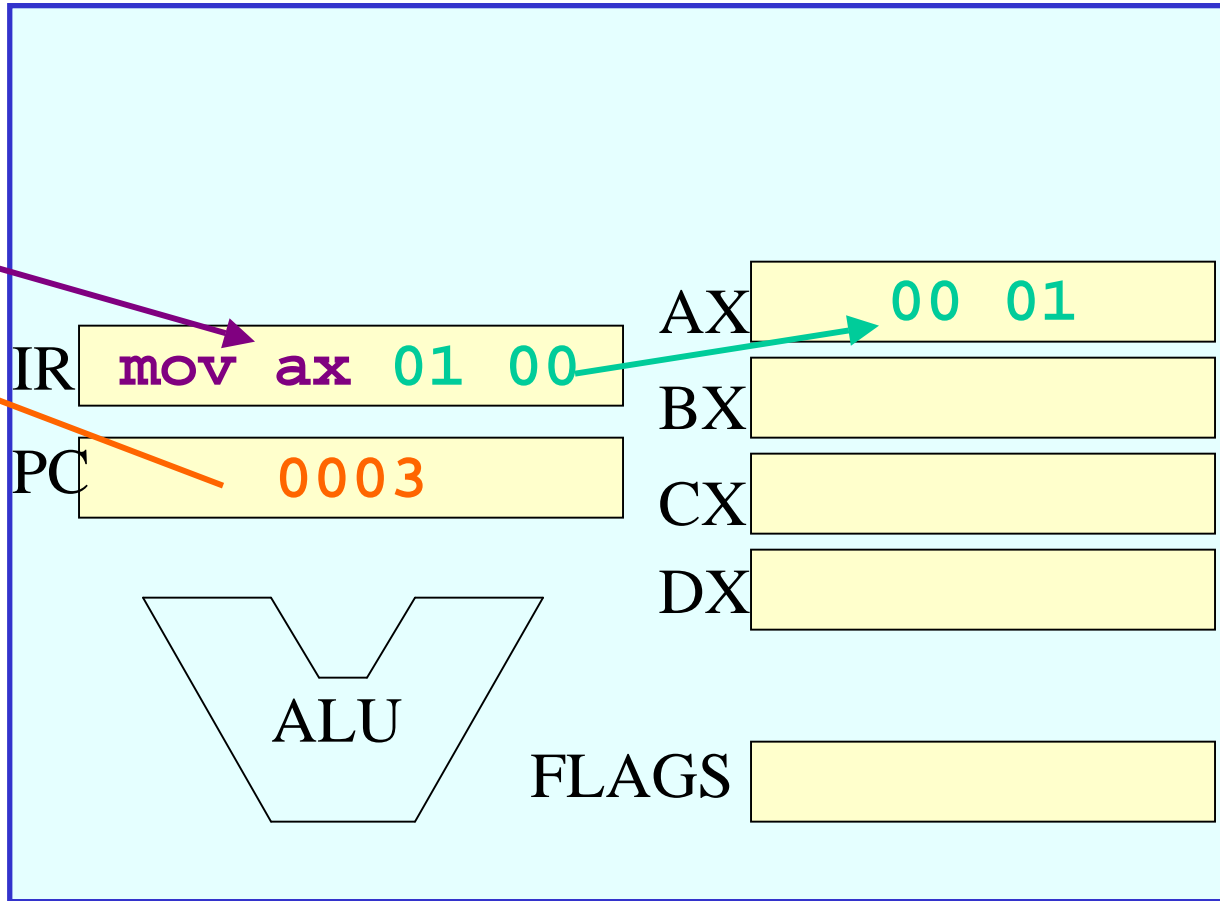
P
R
O
G
R
A
M

Data at location

CPU

Memory

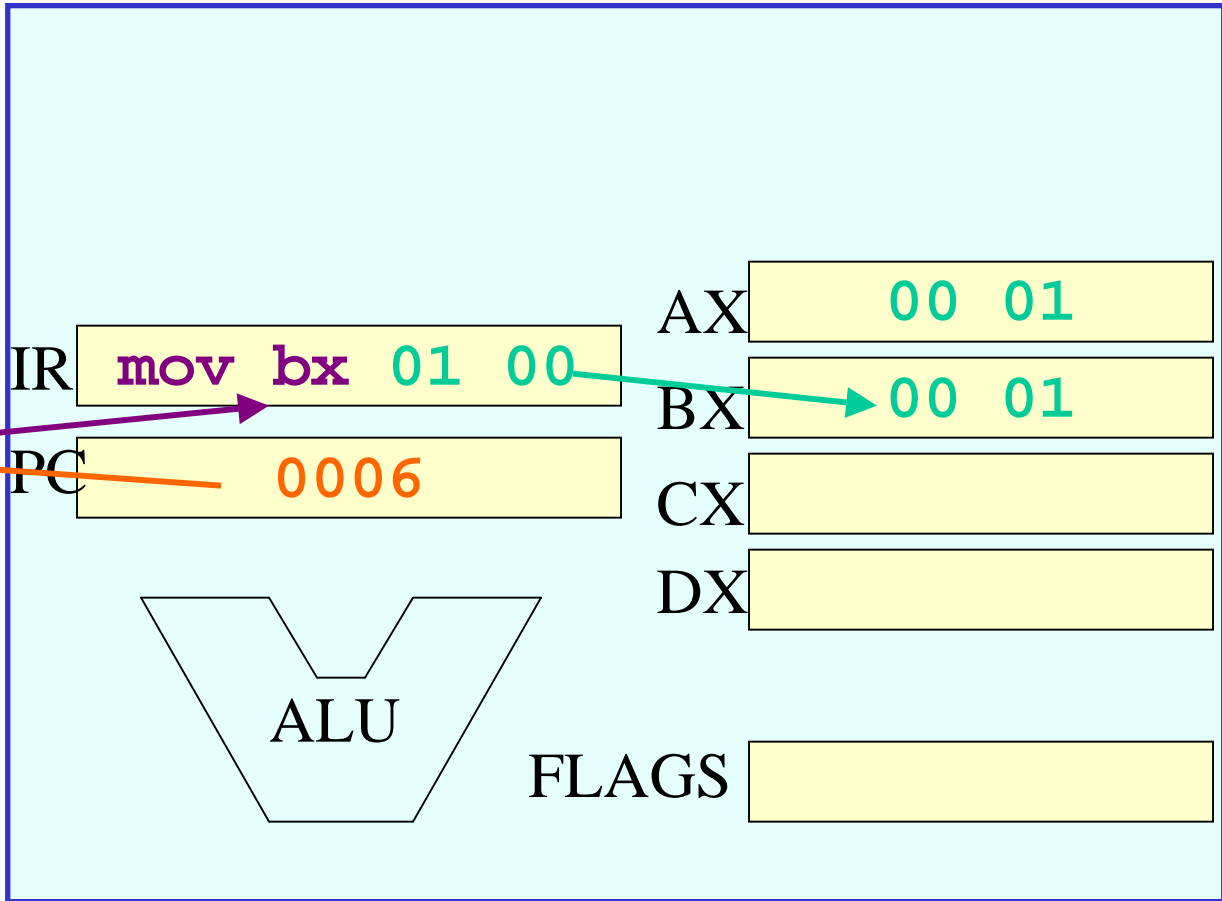
0000	[mov ax]
0001	[01]
0002	[00]
0003	[mov bx]
0004	[01]
0005	[00]
0006	[add ax]
0007	[bx]
0008	[<i>empty</i>]



CPU

Memory

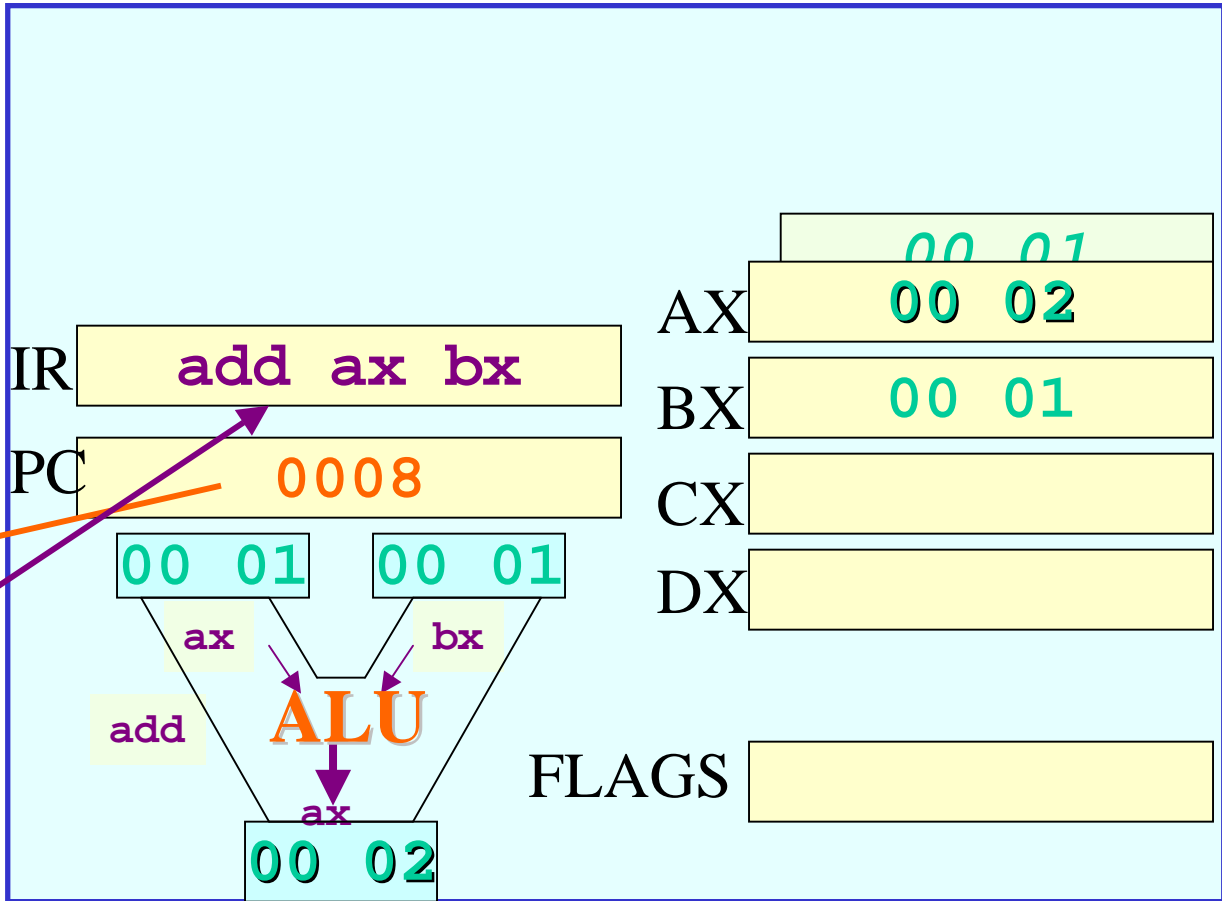
0000	[mov ax]
0001	[01]
0002	[00]
0003	[mov bx]
0004	[01]
0005	[00]
0006	[add ax]
0007	[bx]
0008	[<i>empty</i>]



CPU

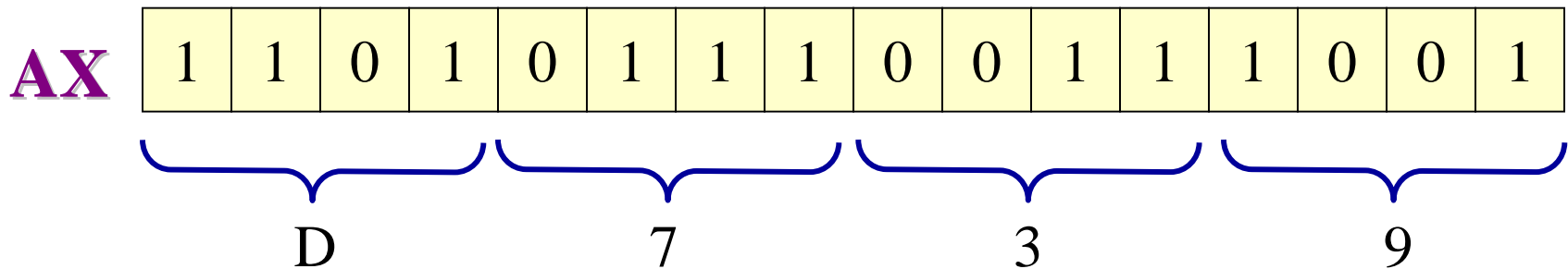
Memory

0000	[mov ax]
0001	[01]
0002	[00]
0003	[mov bx]
0004	[01]
0005	[00]
0006	[add ax]
0007	[bx]
0008	[<i>empty</i>]



Memory Addressing and Registers

- The CPU has 4 General Purpose Registers
 - They are called AX BX CX DX
- Each Register is 16-bits long

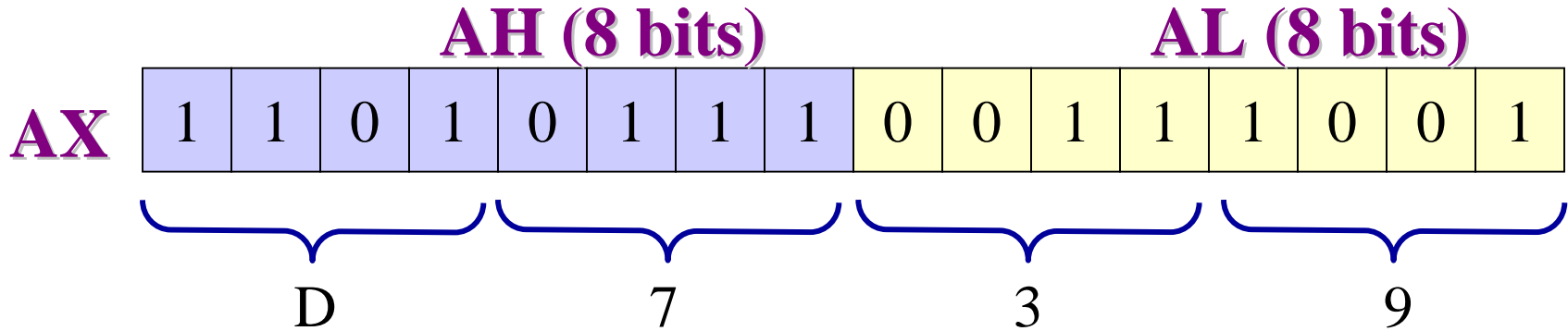


- Each 4 bits holds a Hex Digit so the register holds D739 Hex
- These registers can be accessed 8-bits at a time through their High and Low parts.
- AX has AH (High) and AL (Low), BX has BH and BL, CX has CH and CL, DX has DH and DL.

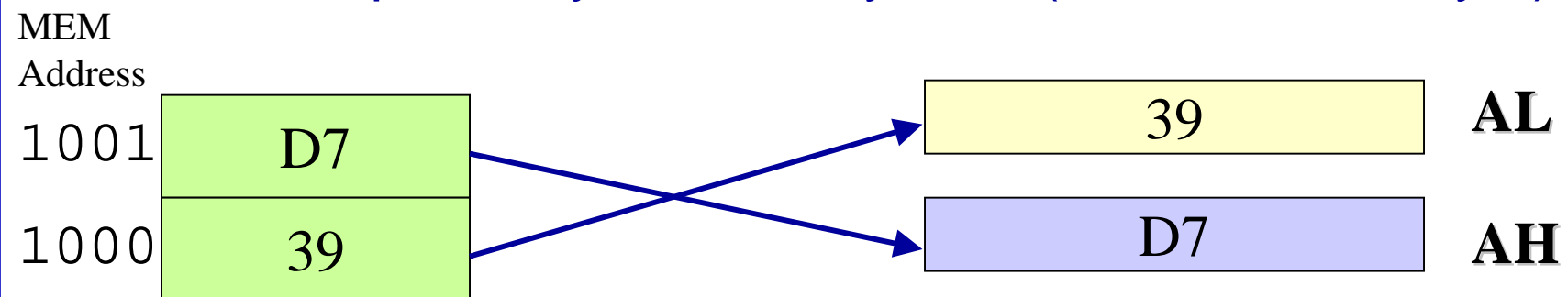
Memory Addressing and Registers

– In the previous example

- **AH** contains D7 and **AL** contains 39



– This maps nicely to Memory Size (8-bits or One Byte)



Moving Data to/from Memory/Registers

– Consider the following:

- MOV AX,03h

– What value is now held in AX ?

?	?	0	3
---	---	---	---

OR

0	0	0	3
---	---	---	---

– The **Size of the Register (AX or AH)** determines whether one byte (8 bits) or two bytes (16 bits) are copied.

– If AX initially is 1234h ...

- then MOV AH, 03h gives
- then MOV AL, 03h gives

1	2	3	4	AX
0	3	3	4	AX
0	3	0	3	AX

Moving Data to/from Memory/Registers

Memory Locations

0008	[1	3]
0007	[A	7]
0006	[2	4]
0005	[2	F]
0004	[C	1]
0003	[9	4]
0002	[B	B]
0001	[4	2]
0000	[2	F]

Watch how the following commands work:

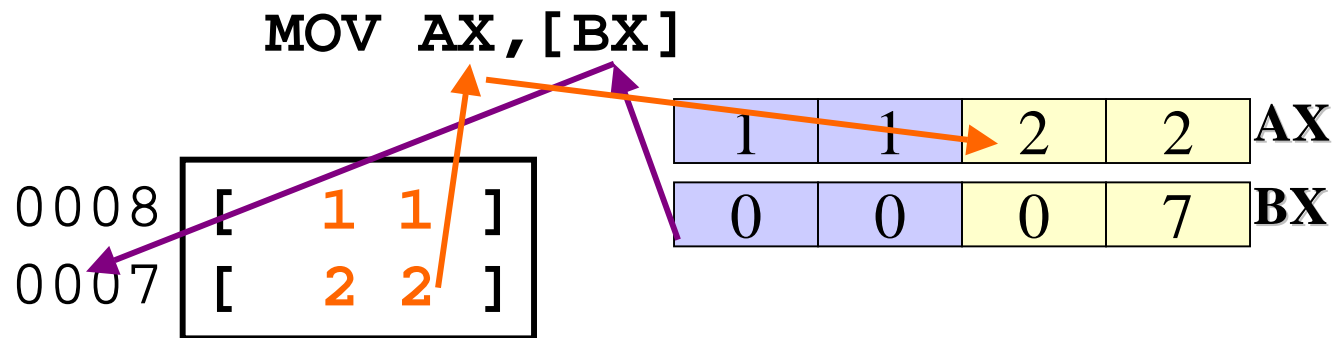
MOV AH, [0008]	1	3	?	?	AX
MOV AL, [0005]	1	3	2	F	AX
ADD AH, AL	4	2	2	F	AX
MOV [0000], AX	4	2	2	F	AX
MOV AX, [0003]	C	1	9	4	AX

Also

MOV BH, AL	C	1	9	4	AX
	9	4	?	?	BX
MOV BL, AH	C	1	9	4	AX
	9	4	C	1	BX
MOV AX, BX	9	4	C	1	AX
	9	4	C	1	BX

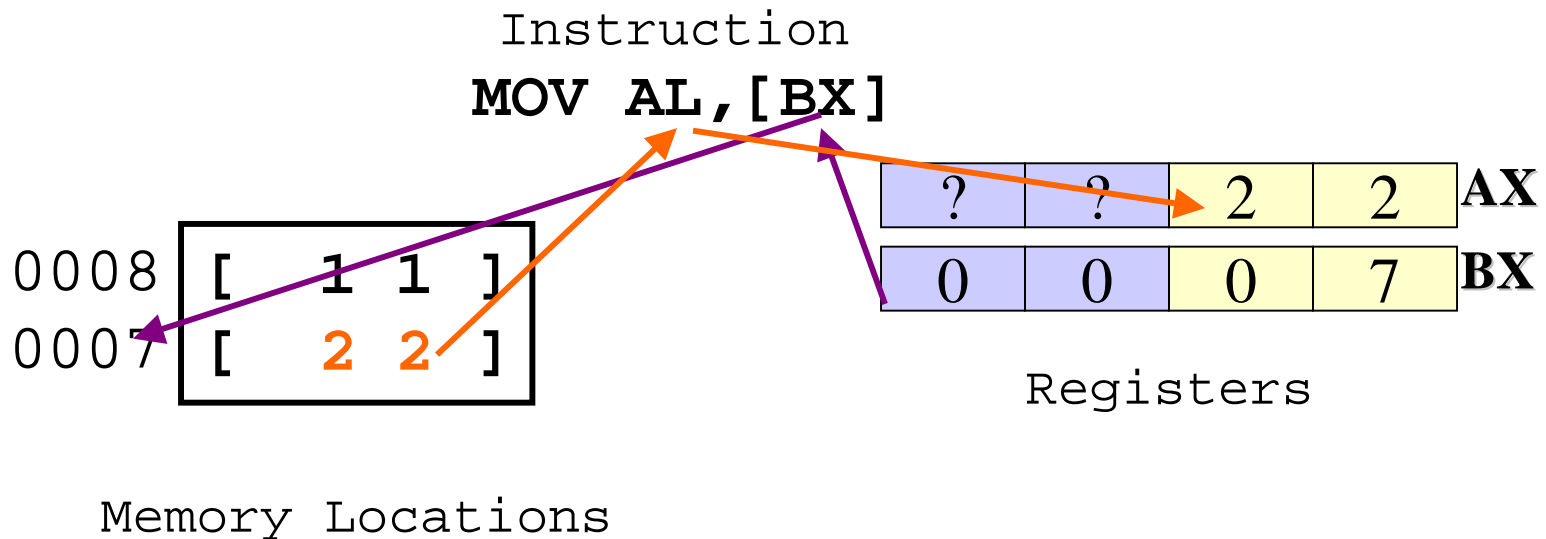
Indexed Addressing

- We know that `MOV AX, [0003]` means copy the contents of memory location 0003/0004 into the AX register.
- What does `MOV AX, [BX]` mean?
- The square brackets tell us that a memory location is being accessed. That means we need the number of the memory location inside the brackets.
- Here we are using the BX register to hold the number of the memory location (Index) that we need.



Indexed Addressing

- **Similarly we can index just one byte at a time**
 - Suppose we want to copy one memory location [0007] into the AL register (Lower half of AX)
 - The instruction is `MOV AL, [BX]` where BX contains the value of the location.



Example

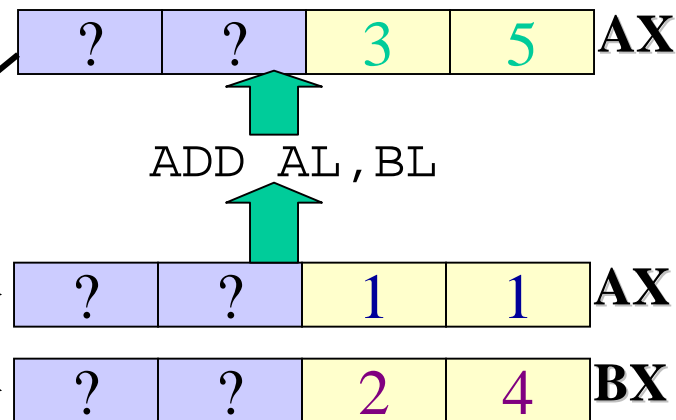
- Write a program that adds the number stored at memory location [0003] to the number stored at memory location [0004] and stores the result in memory location [0005]

Program

```
MOV AL, [0004]
MOV BL, [0003]
ADD AL, BL
MOV [0005], AL
```

0008
0007
0006
0005
0004
0003
0002
0001
0000

[1	3]
[A	7]
[2	4]
[3	5]
[1	1]
[2	4]
[B	B]
[4	C]
[2	F]



Exercises

– For the following initial memory contents diagram where:

0008	[1	3]
0007	[2	7]
0006	[2	4]
0005	[3	A]
0004	[1	7]
0003	[1	4]
0002	[A	B]
0001	[4	0]
0000	[F	F]

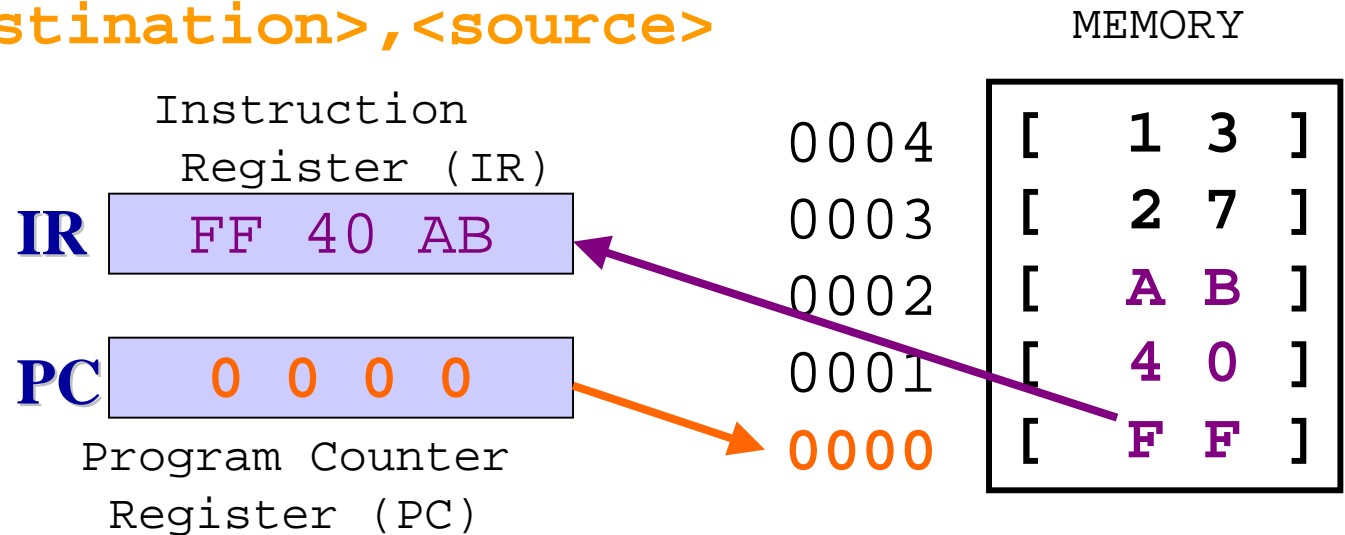
**Initial Memory
Diagram**

- Initially AX BX CX and DX are 0000
- Write down the values in the registers at each stage
 - MOV AH, [0005]
 - MOV CX, [0003]
 - ADD CL, AH
 - MOV AX, CX
 - MOV [0002], AX
 - MOV [0003], CL
 - MOV DX, [0000]
 - MOV BX, DX
 - MOV [0006], BX
- Draw the new memory diagram.

Fetch Execute Cycle

- Each instruction goes through a Fetch phase and an Execute phase.
- Fetching = Copying the instruction stored at the address contained in the PC Register from memory to the Instruction Register.
- Executing = Performing the task specified by the instruction stored in the Instruction Register eg.

copy <destination>, <source>



Fetch Execute Cycle

– Running a program..

