

Name : Tazmeen Afroz

Roll No: 22p-9252

Section: BAI-4A

Coal-lab-task-2

MEMORY:

M1→ MEMORY REPRESENTATION 1:

Memory Representation 1 (M1) typically refers to the memory layout of the emulated environment, here rows represent different segments or areas of memory.

	0	1	2	3	4	5	6	7
1								
DS:0000	CD	20	FF	9F	00	EA	F0	FE
DS:0008	AD	DE	1B	05	C5	06	00	00
DS:0010	18	01	10	01	18	01	92	01
DS:0018	01	01	01	00	02	FF	FF	FF
DS:0020	FF	FF	FF	FF	FF	FF	FF	FF
DS:0028	FF	FF	FF	FF	EB	19	C0	11
DS:0030	A2	01	14	00	18	00	F5	19
DS:0038	FF	FF	FF	FF	00	00	00	00
DS:0040	05	00	00	00	00	00	00	00
DS:0048	00	00	00	00	00	00	00	00

M2 → MEMORY REPRESENTATION 2

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2																
DS:0000	CD	20	FF	9F	00	EA	F0	FE	AD	DE	1B	05	C5	06	00	00
DS:0010	18	01	10	01	18	01	92	01	01	01	01	00	02	FF	FF	FF
DS:0020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	EB	19	C0	11
DS:0030	A2	01	14	00	18	00	F5	19	FF	FF	FF	FF	00	00	00	00
DS:0040	05	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1 Step	2ProcStep	3Retrieve	4Help ON	5BRK Menu	6	7 up										
→ MEMORY REPRESENTATION 2																

REGISTERS:

AX 0000	SI 0000	CS 19F5	IP 0100	Stack +0 0000	Flags 7202
BX 0000	DI 0000	DS 19F5		+2 20CD	
CX 0012	BP 0000	ES 19F5	HS 19F5	+4 9FFF	OF DF IF SF ZF AF PF CF
DX 0000	SP FFFE	SS 19F5	FS 19F5	+6 EA00	0 0 1 0 0 0 0 0

general purpose registers

- **AX** - the accumulator register (divided into AH / AL).
- **BX** - the base address register (divided into BH / BL).
- **CX** - the count register (divided into CH / CL).

- **DX** - the data register (divided into DH / DL).

4 general purpose registers (AX, BX, CX, DX) are made of two separate 8 bit registers, for example if AX= 0011000000111001b, then AH=00110000b and AL=00111001b. therefore, when you modify any of the 8 bit registers 16 bit register is also updated, and vice-versa."H" is for high and "L" is for low part.

SP - stack pointer. Stack Pointer register points to the top of the stack.

IP - the instruction pointer. Instruction Pointer register holds the memory address of the next instruction to be executed.

FLAGS: Flags register contains various status flags such as zero, carry, and overflow.

Flags 7202

Decimal: 7202

Binary: 11100000010010

converting the flag value to binary allows you to examine individual bits and extract information about various settings or states represented by the flag.

Command to watch the memory <memory> <address>

CODE EXPLANATION:

[org 0x0100]

The org command tells the program where to load itself into in memory (ram). In particular this command says to start the program at 100 bytes.

CMD >m1 0100		1	0	1	2	3	4	5	6	7
		DS:0100	B8	05	00	BB	0A	00	01	D8
		DS:0103	BB	0F	00	01	D8	B8	00	4C
0100 B80500 MOV AX,0005		DS:0110	CD	21	EB	04	31	D2	31	C0
0103 BB0A00 MOV BX,000A		DS:0118	89	56	E4	89	46	E6	C7	46

m1 use to view the memory at specific address in memory 1 representation.

AX 0000	SI 0000	CS 19F5	IP 0100	Stack +0 0000	Flags 7202
BX 0000	DI 0000	DS 19F5		+2 20CD	
CX 0012	BP 0000	ES 19F5	HS 19F5	+4 9FFF	OF DF IF SF ZF AF PF CF
DX 0000	SP FFFE	SS 19F5	FS 19F5	+6 EA00	0 0 1 0 0 0 0
CMD >				1 0 1 2 3 4 5 6 7	
				DS:0000 CD 20 FF 9F 00 EA F0 FE	
				DS:0008 AD DE 1B 05 C5 06 00 00	
0100 B80500 MOV AX,0005				DS:0010 18 01 10 01 18 01 92 01	
0103 BB0A00 MOV BX,000A				DS:0018 01 01 01 00 02 FF FF FF	

IP → Instruction pointer stores the address of the next instruction to be executed by the CPU which is 0100 here `mov ax ,5`.

`mov ax, 5`: This line moves the constant value 5 into the 16-bit register ax. Here, ax is being used as a general-purpose register to hold a value.

AX 0005	SI 0000	CS 19F5	IP 0103	Stack +0 0000	Flags 7200
BX 0000	DI 0000	DS 19F5		+2 20CD	
CX 0012	BP 0000	ES 19F5	HS 19F5	+4 9FFF	OF DF IF SF ZF AF PF CF
DX 0000	SP FFFE	SS 19F5	FS 19F5	+6 EA00	0 0 1 0 0 0 0
CMD >				1 0 1 2 3 4 5 6 7	
0100 B80500 MOV AX,0005				DS:0100 B8 05 00 BB 0A 00 01 D8	
0103 BB0A00 MOV BX,000A				DS:0108 BB 0F 00 01 D8 B8 00 4C	
				DS:0110 CD 21 EB 04 31 D2 31 C0	

By pressing F1 first line executes see in AX now 0005 is stored.

IP stores the address of the next instruction to be executed by the CPU which is here 0103 which is `mov bx, 10`.

In this case, MOV instructions typically take up 3 bytes

- 1 for the opcode and 2 for the operand),

the next instruction would be stored at the next memory address after the current instruction.

MOV AX, 5 is stored at memory address 0x00, then MOV BX, 10 would be stored at memory address 0x03 (MOV AX, 5 takes up bytes 0x00, 0x01, and 0x02).

- Data Storage in Memory:


- According to Intel's convention, when storing data like `AX` with a value of `0005h`:

- The LSB (Least Significant Byte) `05` occupies the lower memory address.

- The MSB (Most Significant Byte) `00` occupies the higher memory address.

- Hence, in memory, `AX` with the value `0005h` would be stored as `0500h`.

mov bx, 10: Similarly, this line moves the constant value 10 into the 16-bit register bx.




AX	0005	SI	0000	CS	19F5	IP	0106	Stack	+0	0000	Flags	7200
BX	000A	DI	0000	DS	19F5				+2	20CD		
CX	0012	BP	0000	ES	19F5	HS	19F5		+4	9FFF	OF	DF IF SF ZF AF PF CF
DX	0000	SP	FFFE	SS	19F5	FS	19F5		+6	EA00	0	0 1 0 0 0 0 0
CMD > <input type="text"/>												
1 0 1 2 3 4 5 6 7												
DS:0100 B8 05 00 BB 0A 00 01 D8												
DS:0108 BB 0F 00 01 D8 B8 00 4C												
DS:0110 CD 21 EB 04 31 D2 31 C0												
DS:0118 89 56 F4 89 46 F6 C7 46												
0103	BB0A00		MOV		BX,000A							
0106	01D8		ADD		AX,BX							
0109	BB0F00		MOV		BX,000F							

Now here 0A00 is stored in the BX register.

add ax, bx: This line adds the value stored in register bx to the value stored in register ax and stores the result back in register ax. This instruction effectively performs


$ax = ax + bx$.



AX	000F	SI	0000	CS	19F5	IP	0108	Stack	+0	0000	Flags	7204
BX	000A	DI	0000	DS	19F5				+2	20CD		
CX	0012	BP	0000	ES	19F5	HS	19F5		+4	9FFF	OF	DF IF SF ZF AF PF CF
DX	0000	SP	FFFE	SS	19F5	FS	19F5		+6	EA00	0	0 1 0 0 0 1 0
CMD > <input type="text"/>												
1 0 1 2 3 4 5 6 7												
DS:0100 B8 05 00 BB 0A 00 01 D8												
DS:0108 BB 0F 00 01 D8 B8 00 4C												
DS:0110 CD 21 EB 04 31 D2 31 C0												
DS:0118 89 56 F4 89 46 F6 C7 46												
0106	01D8		ADD		AX,BX							
0108	BB0F00		MOV		BX,000F							

Now the value of ax+ bx stored in ax which is 15.15 in hex is F so 000F is stored in AX.

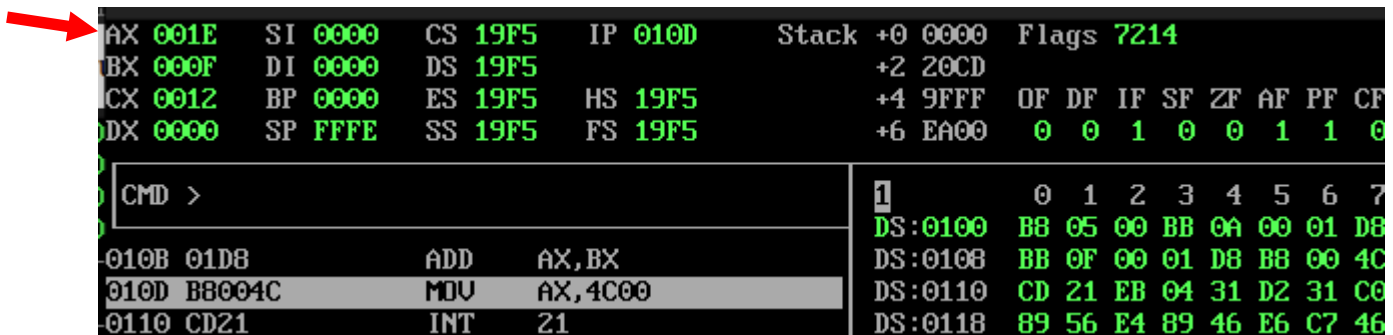
mov bx, 15: This line moves the constant value 15 into register bx, overwriting the previous value.



AX	000F	SI	0000	CS	19F5	IP	010B	Stack	+0	0000	Flags	7204
BX	000F	DI	0000	DS	19F5				+2	20CD		
CX	0012	BP	0000	ES	19F5	HS	19F5		+4	9FFF	OF	DF IF SF ZF AF PF CF
DX	0000	SP	FFFE	SS	19F5	FS	19F5		+6	EA00	0	0 1 0 0 0 1 0
CMD > <input type="text"/>												
1 0 1 2 3 4 5 6 7												
DS:0100 B8 05 00 BB 0A 00 01 D8												
DS:0108 BB 0F 00 01 D8 B8 00 4C												
DS:0110 CD 21 EB 04 31 D2 31 C0												
DS:0118 89 56 F4 89 46 F6 C7 46												
0108	BB0F00		MOV		BX,000F							
010B	01D8		ADD		AX,BX							
010D	BB0F4C		MOV		AX,4C00							

Now value 15 (000F) is stored in the BX register.

add ax, bx: Again, this line adds the value stored in register bx (which is now 15) to the value stored in register ax and stores the result back in register ax.(30)



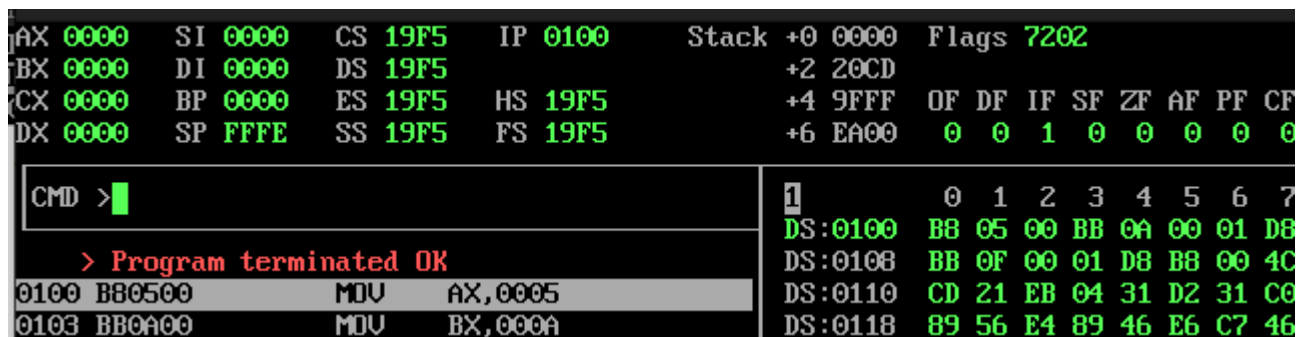
```
AX 001E SI 0000 CS 19F5 IP 010D Stack +0 0000 Flags 7214
BX 000F DI 0000 DS 19F5      +2 20CD
CX 0012 BP 0000 ES 19F5 HS 19F5  +4 9FFF OF DF IF SF ZF AF PF CF
DX 0000 SP FFFE SS 19F5 FS 19F5  +6 EA00  0 0 1 0 0 1 1 0

CMD >
010B 01D8      ADD    AX,BX
010D B8004C     MOV    AX,4C00
0110 CD21       INT    21
DS:0100 B8 05 00 BB 0A 00 01 D8
DS:0108 BB 0F 00 01 D8 B8 00 4C
DS:0110 CD 21 EB 04 31 D2 31 C0
DS:0118 89 56 E4 89 46 E6 C7 46
```

Now the value of ax+ bx stored in ax which is 30.30 in hex is 1E so 001E is stored in AX.

Our application relies on the operating system for all tasks. Occasionally, the OS interrupts our execution and assigns CPU tasks.

- Interrupt 0x21: This interruption prompts the CPU to perform specific work.
- Exit Operation:- Tasked with the work code 4C00, which signifies program termination.



```
AX 0000 SI 0000 CS 19F5 IP 0100 Stack +0 0000 Flags 7202
BX 0000 DI 0000 DS 19F5      +2 20CD
CX 0000 BP 0000 ES 19F5 HS 19F5  +4 9FFF OF DF IF SF ZF AF PF CF
DX 0000 SP FFFE SS 19F5 FS 19F5  +6 EA00  0 0 1 0 0 0 0 0

CMD >
> Program terminated OK
0100 B80500     MOV    AX,0005
0103 B80A00     MOV    BX,000A
DS:0100 B8 05 00 BB 0A 00 01 D8
DS:0108 BB 0F 00 01 D8 B8 00 4C
DS:0110 CD 21 EB 04 31 D2 31 C0
DS:0118 89 56 E4 89 46 E6 C7 46
```

Program Terminated.