1	2	3	4	5	6	7	Total

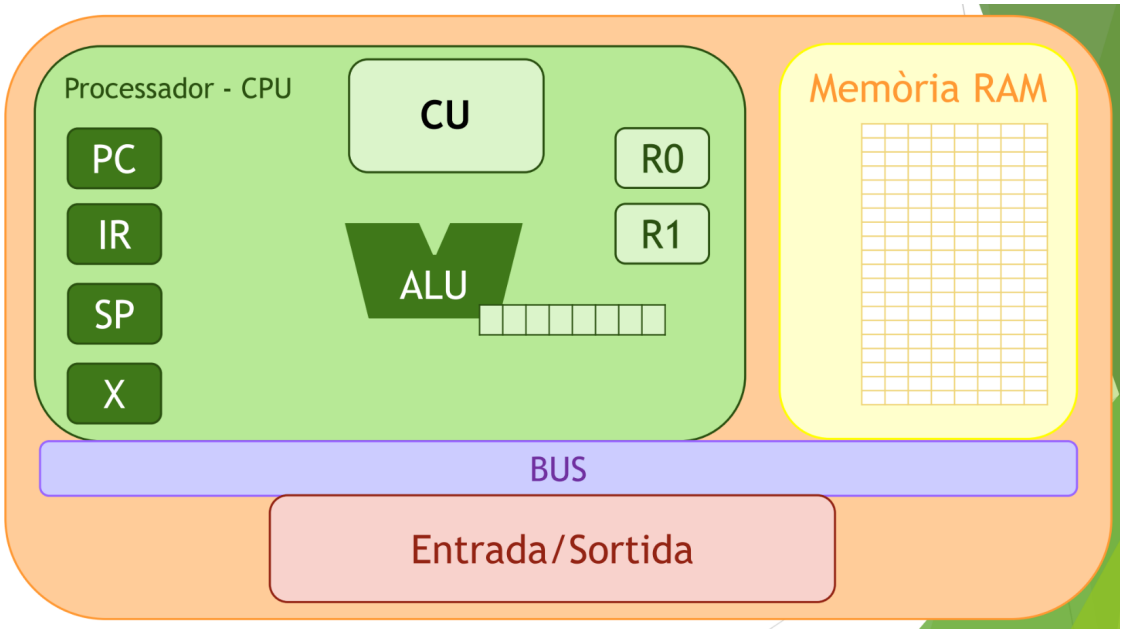
GIA IA-UAB 31 de marzo de 2023

Computer Fundamentals

Surname:

Name: DNI/NIU:

1.- [1 p.] Describe briefly the main parts of a computer and their function



CPU:

- Control Unit
- ALU
- PC, IR, SP
- Registers

BUS
Memory
BUS
Input/Output system

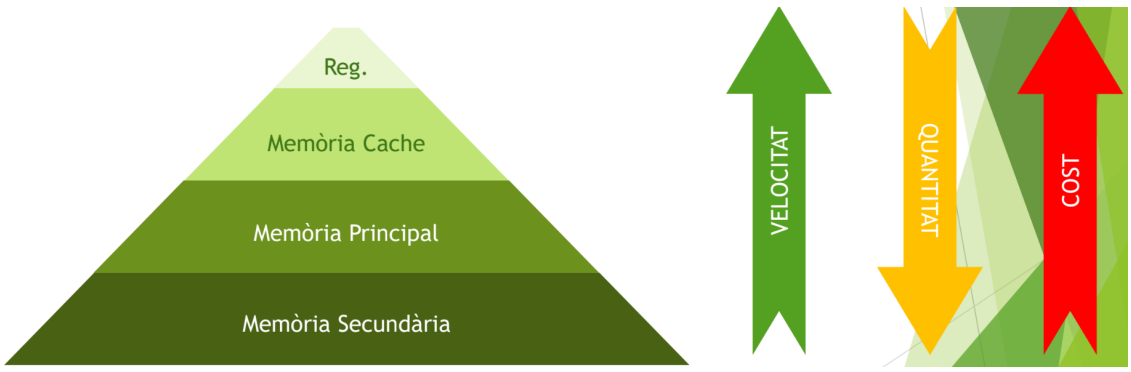
2. -[1 p.] Describe how a computer manages the process of executing an instruction of a program

Fetch: The control unit requests instructions from the main memory that is stored at a memory's location as indicated by the program counter (instruction counter)

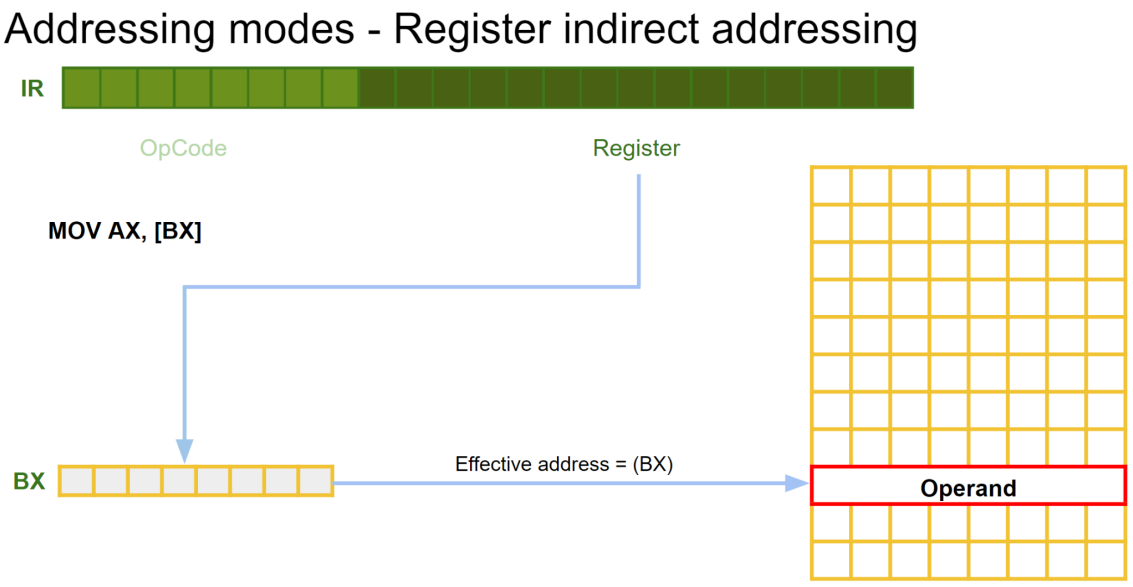
Decode: Received instructions are decoded in the instruction register. This involves breaking the operand field into its components based on the instruction's operation code (opcode)

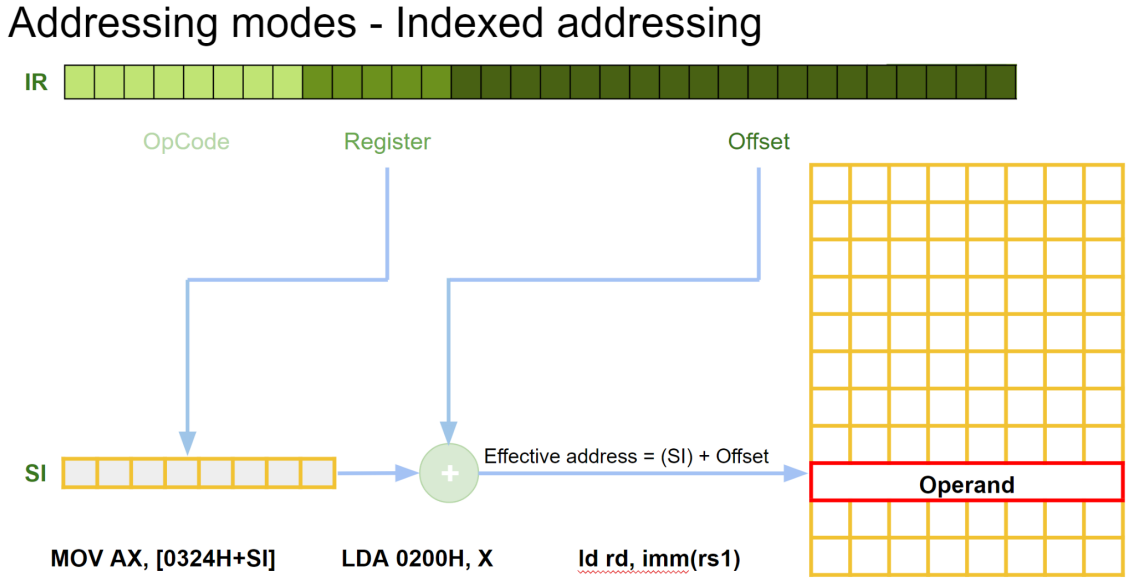
Execute: This involves the instruction's opcode as it specifies the CPU operation required. The program counter indicates the instruction sequence for computer. These instructions are arranged into the instructions register and as each are executed, it increments the program counter so that the next instruction is stored in memory. Appropriate circuitry is then activated to perform the requested task. As soon as instructions have been executed, it restarts the machine cycle that begins the fetch step

3. -[1 p.] Describe the memory hierarchy of a computer: types of memory, use of each level, characteristics of the hierarchy



4. -[1 p.] Describe the indirect and indexed addressing modes: how they work using the computer resources they need to get to an operand using an example





5. -[2 p.]Consider three 4x4 arrays (A, B and C) of 4 bytes elements. Given the following high-level code that performs the product of matrices A by B and generate array C:

```
int A[4][4], B[4][4], C[4][4];
int i, j, k;
for (i=0; i <4; i++)
{
   for (j=0; j <4; j++)
   {
      C[i][j]=0;
      for (k=0; k <4; k++)
         C[i][j]=C[i][j]+A[i][k]*B[k][j];
   }
}
```

complete the following code in assembler:

```
INI:        MOV ESI, 0
BUCLE_i:    CMP ESI, 4
            JGE FI_i
            MOV EDI, 0
BUCLE_j:    CMP EDI, 4
            JGE FI_j
            CALL Calc_Index
            MOV [Position], EBX
            MOV C[EBX],0
            MOV ECX, 0
BUCLE_k:    CMP ECX, 4
            JGE Fi_k
            PUSH EDI
            MOV EDI, ECX
            CALL Calc_Index
            POP EDI
            MOV EAX, A[EBX]
            PUSH ESI
            MOV ESI, ECX
            CALL Calc_Index
            POP ESI
            MOV EDX, B[EBX]
            MUL EDX
            MOV EBX, [Position]
            ADD C[EBX], EAX
            INC ECX
            JMP BUCLE_k
FI_k:       INC  EDI
            JMP BUCLE_j
FI_j:       INC ESI
            JMP BUCLE_i
FI_i:       END
Calc_Index: PUSH EAX
            MOV EBX, ESI
            SHL EBX, 4
            MOV EAX,EDI
            SHL EAX, 2
            ADD EBX, EAX
            POP EAX
            RET
```

6. -[2 p.] Perform the following operations using 8-bit binary numbers in two's complement. Which flags would be activated? Will the results be correct? Why?

```
    117            -58
+  - 58          + -77
________         ______
```

```
       117          1     1
+      -58          01110101
                    11000110
   _________       __________
        59        1 00111011      +59  CARRY
```

```
       -58        1       11
+      -77          11000110
                    10110011
   _________       __________
      -135        1 00111001      +57  CARRY i OVERFLOW
                                  Resultat incorrecte
```

7. - [2 p.] Given the following assembler program fragment:

```
1 MOV SI, #2
2 MOV AL, 31
3 MOV 20, AL
4 ADD AL, 20[SI]
5 MOV (31), AL
6 CMP (12), AL
7 JG 4
8 MOV 20[SI], AL
```

where the addressing modes used are:

n Direct
#n Immediate
n[X] Indexed
(n) Indirect

Show in the following table the evolution of the records and the positions of memory, indicating the new values that change and the operands that receive a read operation.

	Registers		**Memory**								
	AL	**SI**	**10**	**11**	**12**	**20**	**21**	**22**	**30**	**31**	**32**
INSTRUCTIONS	0	0	30	20	10	1	2	5	10	20	30
MOV SI,#2		2									
MOV AL,31	20									R	
MOV 20,AL						20					
ADD AL,20[SI]	25							R			
MOV (31),AL						25				R	
CMP (12),AL			R		R						
JG 4											
ADD AL,20[SI]	30							R			
MOV (31),AL						30				R	
CMP (12),AL			R		R						
JG 4											
MOV 20[SI],AL								30			