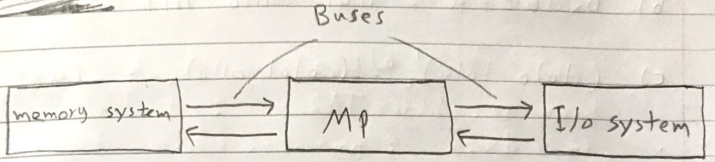
Mahmoud ALKASEM
18290636

1) a)



b) 1. Data transfer between itself and the memory or I/O system.
2. simple arithmetic and logic instruction.
3. program flow via simple decisions.

c) 1. Data Bus: transfer information between MP and its memory and I/O system.
2. Control Bus: contains lines that select the memory or I/O and cause them to perform read or write operation.
3. Address Bus: allow The MP to send the address to memory.

Scanned with CamScanner

2) a) $(10111.1101)_2 = (11.8125)_{10}$

b) 0x DEFA = 1101 1110 1111 1010

c) $(1010\ 1010)_2 = -(0101\ 0110)_2 = -(86)_{10}$

3)

a) CS : code segment
IP : Instruction pointer

b) SS: stack segment
SP: stack pointer

c) SI: Source Index used for LODS instruction
DI: Destination Index used for stos instruction

d) Ax register and Dx register

4)

interrupt: is either a hardware-generated CALL or a software-generated call

5)

It allows Dos programs to be relocated in the memory system, and it allows programs written to function in the real mode to operate in the protected mode system.

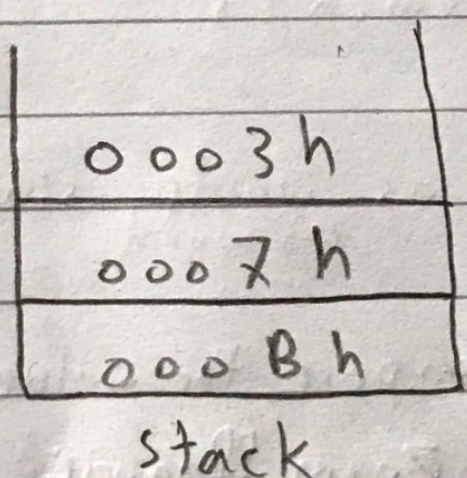
Scanned with CamScanner

6)

a) AL = 1010 0011
CF = 1

b) AL = 0110 1000
CF = 1

6)



Scanned with CamScanner

7) a)

```
TABLE DW  ......          ; table store the numbers
                          ; as world

cld            ;  auto - increment mode
mov si, offset TABLE    ; load effective address of table
mov dx, 0      ; intialize counter to zero
; loop through all elements

IO: lodsw      ; loads as words since they're stored
                  as word

cmp ax, 0xFFAA ; compare the value from ax with this value
jz   finish          ; if zero flag is one stop iteration
shr ax, 1  ; shift right ax register by 1
               ; so least significant bit copied to Cf
jnc IO        ; if Cf is zero then number is even
              ; and move to next element
inc dx        ; if number is not even then it's odd
              ; so I increment dx by 1

finish:               ; stop the program
      hlt
```

Scanned with CamScanner

b)

```
polynomial  PROC
        mov   dx, 0          initialize dx to zero
        mov   cx, 4          push 4 to cx register
        mov   bx, ax         push ax into bx

Loop1:

        CALL  power          ; call power function
        push  cx             ; push cx again to stack
        Loop2:
              add dx, ax     ; add term to dx
              Loop Loop2     ; loop
        pop cx               ; get value from stack to cx
        Loop Loop1           ; loop
    RET                      ; return address of next instruction
polynomial ENDP              ; end procedure


; power function calculate the x value to power of n
power   PROC
    Push cx          ; first push cx to stack to not loose it
    mov ax, 1        ; set ax value to 1 for multiply
    L0:
        IMUL bL      ; multiply al by bl and store in ax
        Loop L0      ; Loop as counter number
    pop cx           ; get the value from stack to cx
RET                  ; return address of next instruction
power ENDP           ; end procedure
```

Scanned with CamScanner