DHARAMSINH DESAI UNIVERSITY, NADIAD

Faculty of Technology

Online Sessional examination

B.Tech (CE) sem : 5th

subject : Microprocessor Fundamental & Programming

Roll no. CE 136 Date : 23rd Oct 2020

signature : D.R.Vagh Time : 10:00 am

Total Pages : 11

Q.1

(a) MOV A, #65H

(d) 1. DW (Define word) for word integer
2. DD (Define double word) for short real (single precision) & short integer
3. DQ (Define quad word) for long real (double precision) & for long integer
4. DT (Define ten bytes) for Packed decimal & for temporary real.

(g) - The assembler insert before code for most of the 8087 instruction is ESC (11011) to the HSB PART.
- So, when the fetching instruction is given by the 8086 microprocessor the instruction is received by both 8086 and 8087. So while decoding it 8087 can able to it is a 8087 instruction (opcode) and 8086 will remove that opcode.

(e) LARGE and HUGE memory models should be used in calling a far processor.

~~beez~~ ~~for~~

because for very long programs that require more that one code segment and operate on large amounts of data which would require more than one data segment. so the LARGE and HUGE models are most appropriate.

(b) Whenever the coprocessor is RESET, the top of the stack register is set to 07

(c)

```
        mov     R0 , #08H
        SETB    P1.7
        MOV     R1 , #20H
L1 :    MOV     C, P1.7
        RRC     A
        DJNZ    R0, L1
        MOV     @R1, A
        END
```

(f) - Auto initialization mode is used to initialize the transfer again with same count and same base address from the same memory area.

- Registers affected BA, BWC base register, base word count,

DATE
PAGE No

Q.2

(a) - Each 8086 and 8087 gets a copy of the instruction as they are fetched from memory.

Since, all the instructions of the 8087 have 9BH in the most significant byte of the opcode, the 8088/86 ignores these instructions.

- 9BH is the opcode for the 8086 ESCAPE instruction.
- The 8087 ignores any opcode that lacks 9BH.
- It must be made clear that although both receive a copy of each device ~~that~~ fetched opcode, only the 8086 can fetch opcodes since it is the only device that has the instruction pointer.
- how the 8086 makes sure it is not flooding the 8087 by fetching instructions for the coprocessor faster than the 8087 can process them.

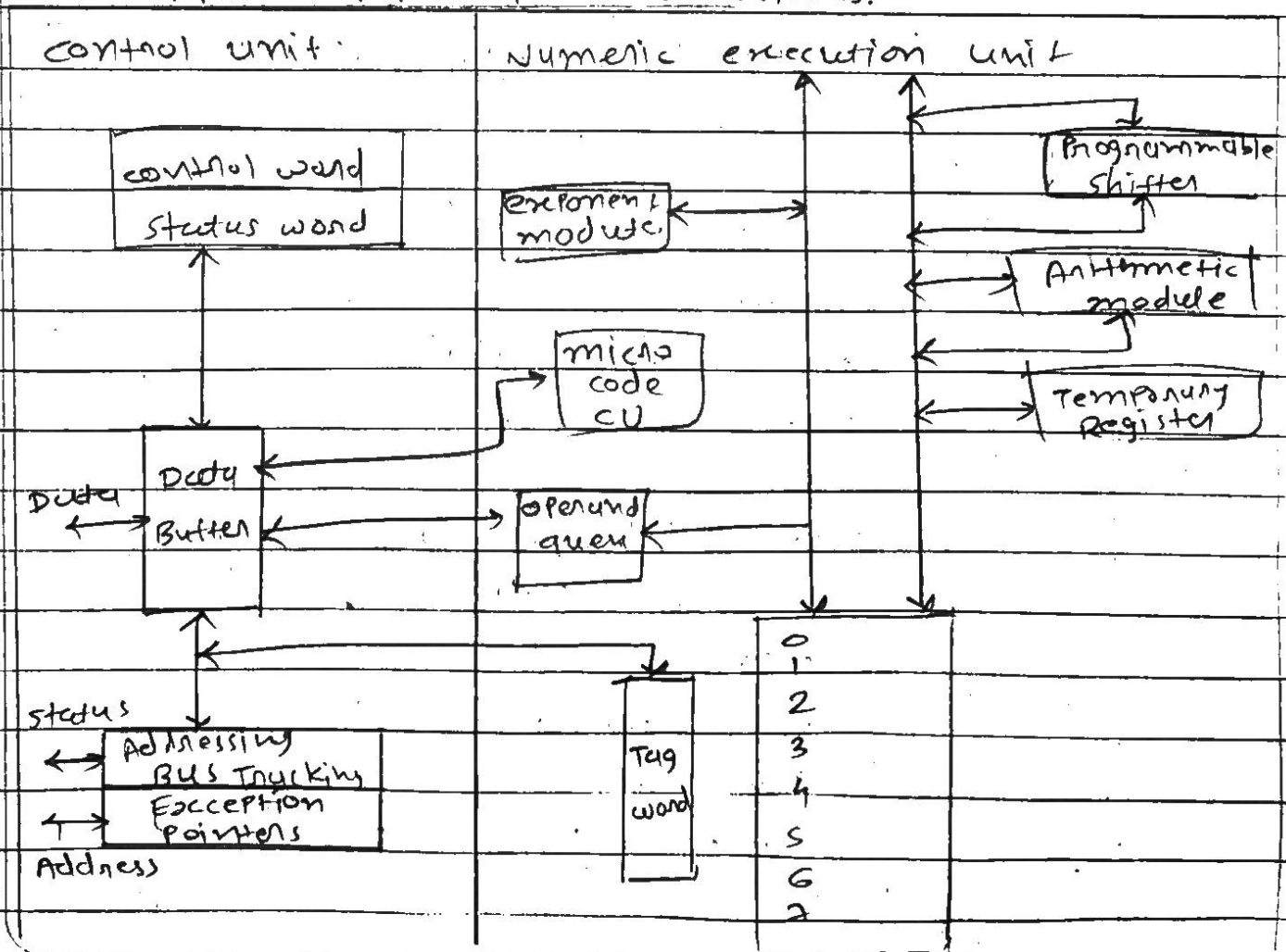
→ The first rule of working together is that the 8086 cannot fetch another 8087 instruction until the 8087 has finised execution of the present instruction.

- when the 8087 is executing an instruction, it activates the BUSY pin automatically by putting high on it.
- This pin is connected to the 'TEST' pin of the 8086.
- next, the 8086 fetches the next instruction which is a WAIT instruction that has been inserted by the assembler, and executes it, thereby going into an internal loop while continuoily monitoring the 'TEST' pin to see when this pin goes low.
- when the 8087 finishes execution of the present instruction, it pulls down (low) the Ready pin, indicating through the TEST' pin to the 8086 that it can now send the next instruction to the 8087.

DATE
PAGE No.

→ Architecture of 8087 coprocessor

- it designed to operate with 8086 microprocessor.
- They both can execute their respective instruction simultaneously.
- microprocessor interprets and executes the normal instruction set and 8087 interprets and executes only the coprocesor instructions.



Q.2

(c)

```
data segment
    student struc
        id db ?
        Rollno db ?
        Name db 12 dup(?)
        semester db ?
    student ends

    st student 5 dup (<11,111, 'Divya', 5>)

data ends
code segment
    assume cs:code, ds:data
    mov ax, data
    mov ds, ax

    mov bl, st[2].semester
    or bl, 30H
    mov dh, 2
    mov dl, bl
    mov ah, 2
    int 21h
code ends
end
```

DATE
PAGE NO

Q.3

(b)

```
data segment
    r1 dd 5.0
    four dd 4.0
    three dd 3.0
    cube dd ?
    volume dd ?
data ends

code segment
    assume cs: code , ds: data
    mov ax, data
    mov ds, ax

    finit
    fld r1          ; st(0) = r1
    fmul st, st(0)  ; st(0) = r1^2
    fld r1          ; st(0) = r1, st(1) = r1^3
    fmul            ; st(0) = r1^3
    fldPi           ; st(0) = π, st(1) = r1^3
    fmul            ; st(0) = π r1^3
    fld four        ; st(0) = 4
                    st(1) = π r1^3
```

```
fmul        ; st(0) = 4Π×r^3
fld three   ; st(0) = 3 , st(1) = 4Π×r^3
fdiv        ; st(0) = 4Π×r^3 / 3
fst volume

    int 03
code ends
end
```

DATE
PAGE KO

Q.3

(a) - After receiving a request from Peripheral Device through DREQ signal, The DMA Controller will give HOLD signal to the microprocessor.

- The sent HOLD / HRQ signal shows interest of DMA controller to become bus master for transfering data from I/O device to memory.
- on receiving HOLD Request, microprocessor relinquing control over the Buses and then it sends HLDA signal to DMA controller.
- Now, as Bus is DMA can hold the control of Bus after recieving HLDA signal. Thus action cycle of DMA controller is enabled now.
- now, DMA controller sends DAck signal to controlled of I/O device, indicating it is ready to do the data transfer.
- Then the Data transfer begins and these steps are followed by DMA controller after receiving request from peripheral device.