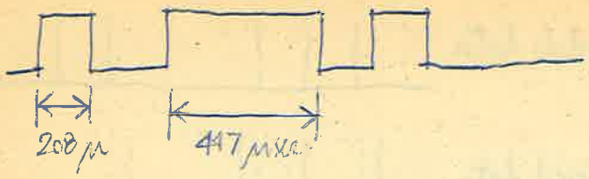
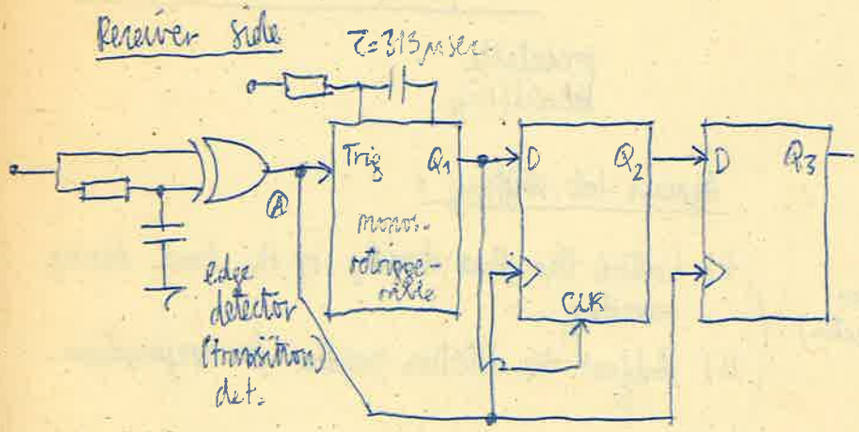
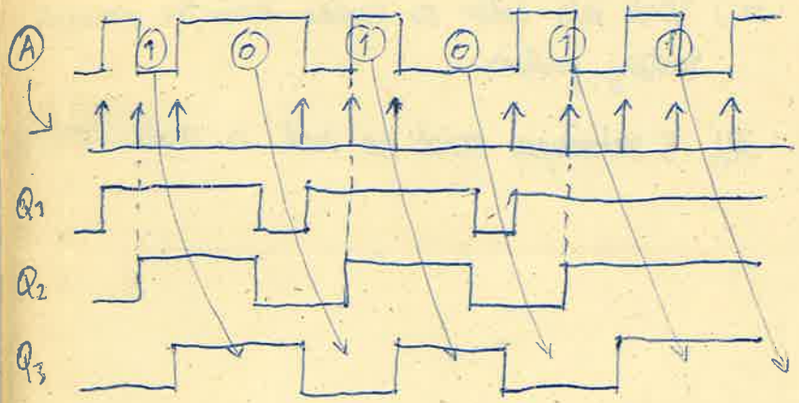
541 2/1/84

Tape Interface



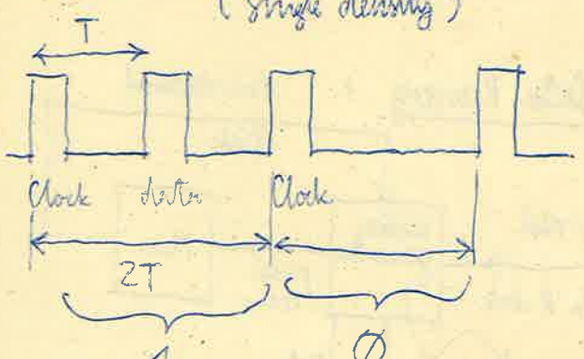
Receiver side





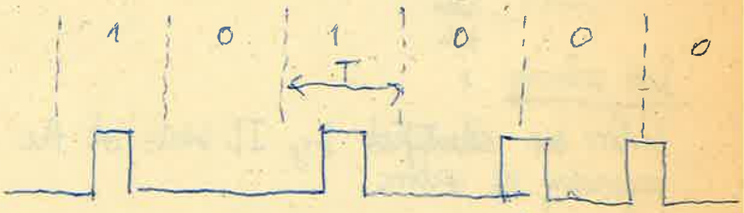
Magnetic disk recording

→ FM recording (Manchester encoding)
(Single density)



	8"	5¼"
T	2 μsec	4 μsec
Byte	32 μsec	64 μsec

MFM (Modified FM) (double density)



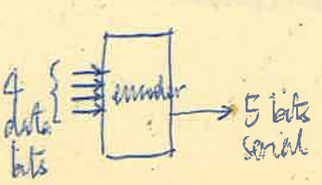
	8"	5¼"
T	2 μsec	4 μsec
Byte	16 μsec	32 μsec

If two neighbours are zeros, put a pulse in between.
(to maintain synchronicity)

logic 1 : Flux transition in the middle of the window.
logic 0 : No transition in that window but flux transition between two.

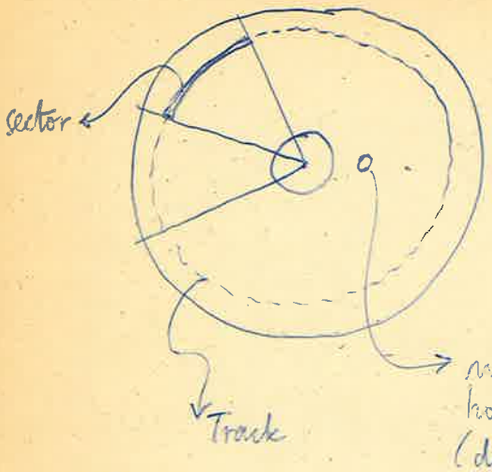
GCR (Group Code Recording) :

Four data bits are encoded as a block of 5 code bits. 1's recorded as flux transitions 0's as no transitions. No clock transitions



Every block has at least one transition

Tracks and Sectors :

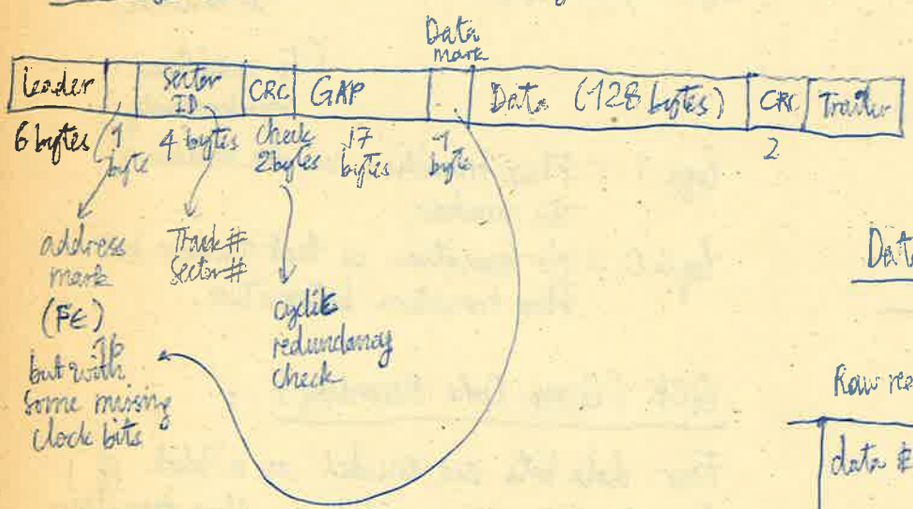


Hard sectoring :
Sectors are identified by extra holes.

Soft sectoring :
Sectors are identified by ID blocks at the beginning of sectors.

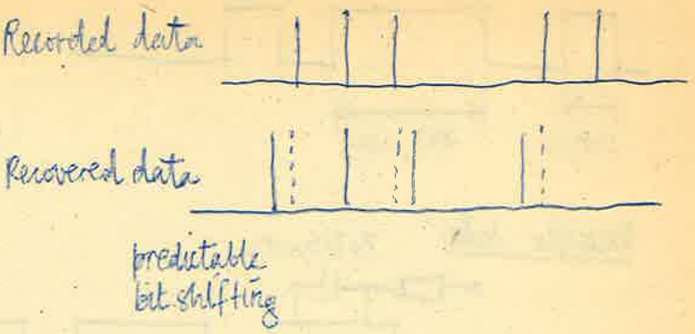
# of tracks	8"	5 1/4"
	77	(35) 40 - 80 - 160

Sector format : (Single Density)



Leader, Gap & trailer are there to provide synchronization.

Magnetic disk recording Bit shifting

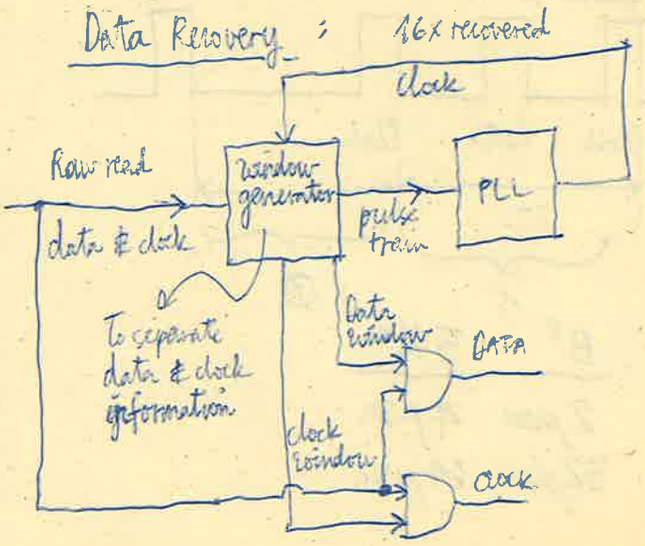


Against bit shifting :

i) Control the flux density of the head during recording.
ii) Adjust the detection window for compensation
→ in floppy disk drive
iii) Write bits later or earlier than the normal writing position.

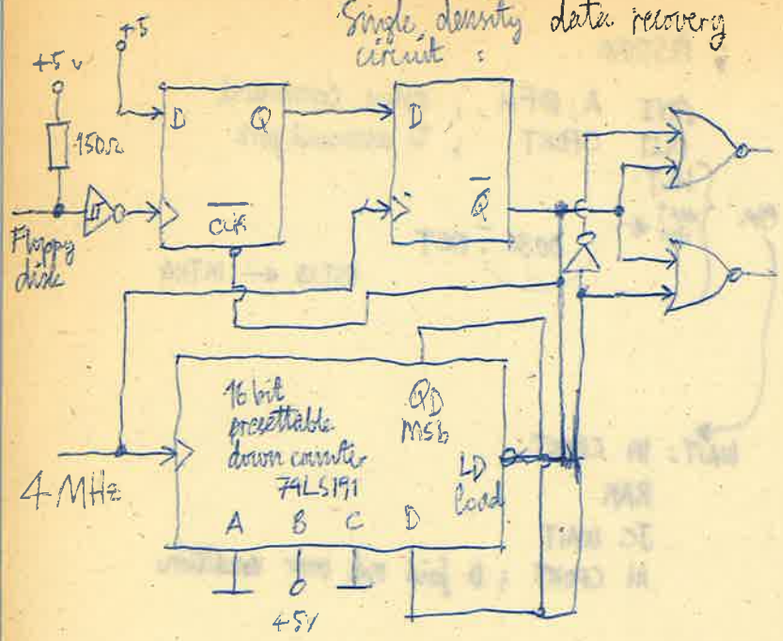
All 3 techniques must be used in double density

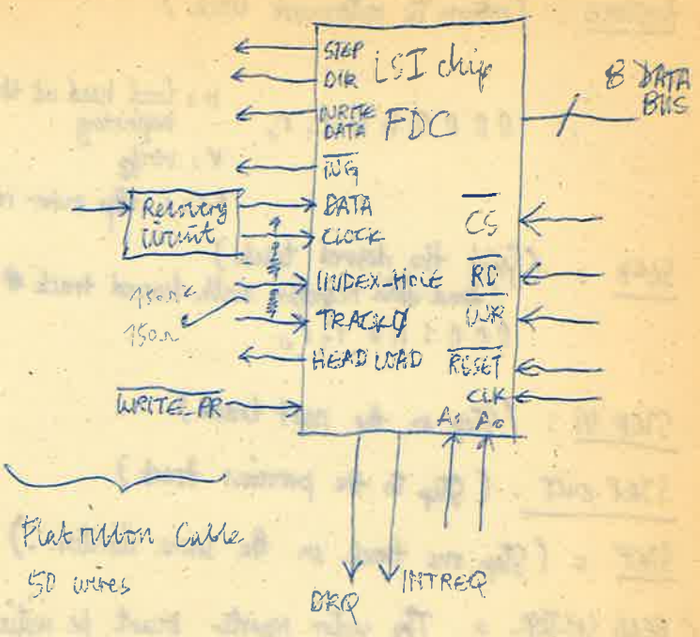
Data Recovery :



8 counts data window
8 counts clock window

Single density data recovery circuit :

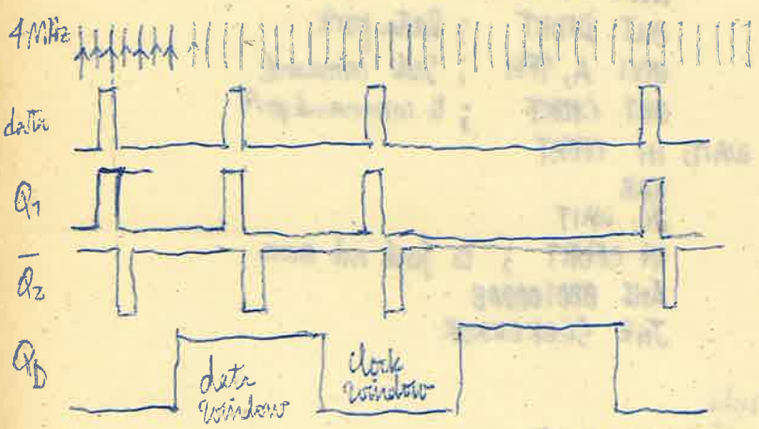




6/1/84

LSI Floppy Controller Chips :

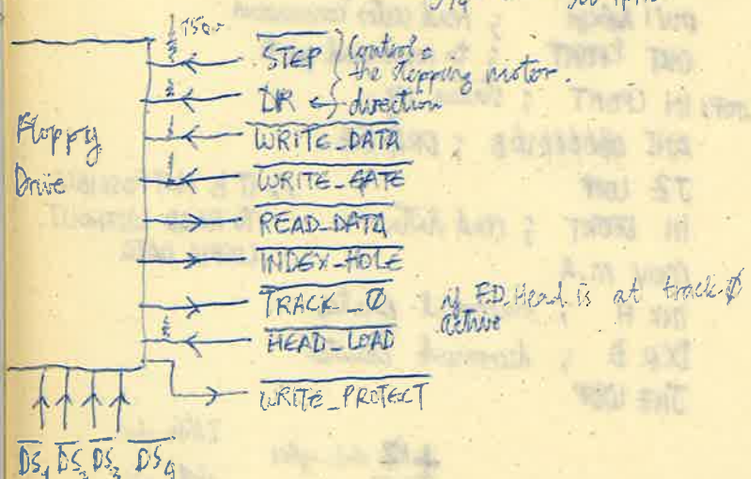
Western Digital 1771 1791
Motorola 6843
Intel 8271
NEC 372,765 ← double density

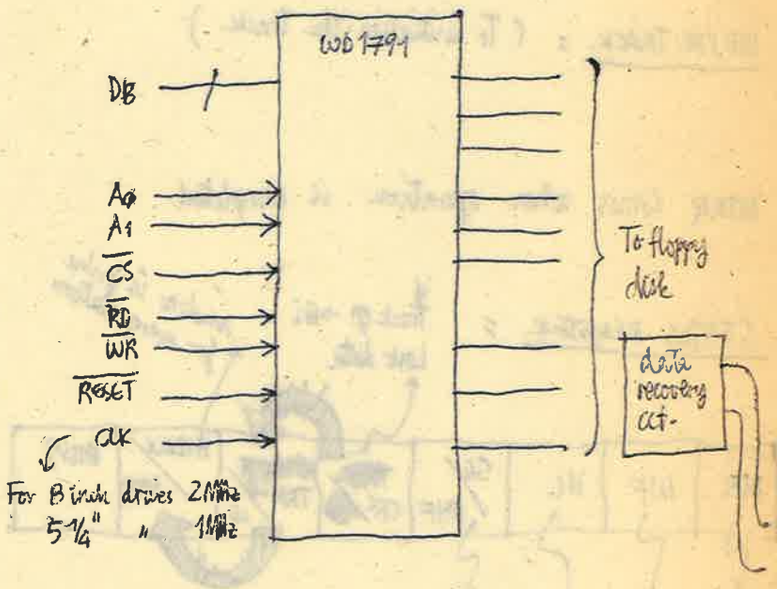


WESTERN DIGITAL 1791

Floppy Disk Drive Interface :

8" drive 360 rpm
5¼" " 300 rpm





Four registers ; A1 A0 ←

Command / Status Register	0 0	: to issue commands, to receive status.
Track register	0 1	: holds track # (0, ... n)
Sector register	1 0	: holds sector # (1, ... n)
Data register	1 1	: holds read data, data to be written ....

IBM standard 3740 (8") 77 tracks, 26 sectors/track
→ 128 bytes/sector → 256 bytes/sector

Commands that can be issued :

RESTORE : (return to outermost track)

0 0 0 0 h V r_1 r_0

h : load head at the beginning
V : verify
$r_1 r_0$: step motor rate

SEEK : (find the desired track)
(load data register with desired track #)
0 0 0 1 h V r_1 r_0

STEP IN : (Step in the next track)

STEP OUT : (Step to the previous track)

STEP : (Step one track in the same direction.)

READ SECTOR : 10000100 — only data register changes. The sector register must be initialized it loads the head & presents the data one at a time at the data register.

WRITE SECTOR : The sector register must be loaded awaits data one at a time in data register.

DRQ line shows the time data is ready or is to be given.

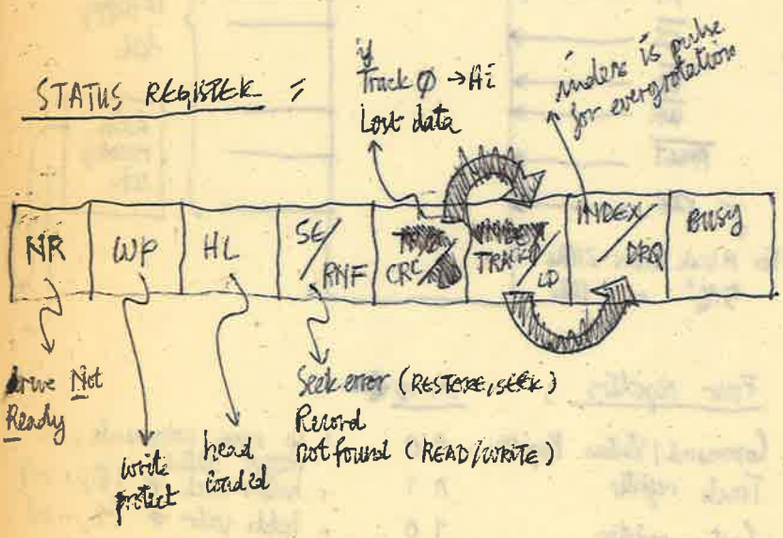
READ ADDRESS : (To read track & sector #)

READ TRACK : (To read the whole track including trailer, gap, leader,.)

WRITE TRACK : (To initialize the track)

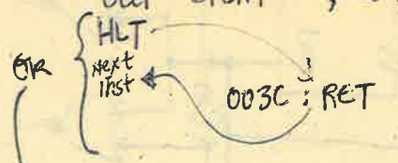
INTRQ comes when operation is completed.

STATUS REGISTER :



Software interface of LSI FDC :

```
; RESTORE
    MVI  A, 0FH   ; restore command
    OUT  CPORT    ; to command port
```




```
    MVI  A, TRACK #
    OUT  DPORT    ; Data port
    MVI  A, 1FH   ; Seek command
    OUT  CPORT    ; to command port
WAIT: IN  CPORT
    RAR
    JC  WAIT
    IN  CPORT     ; to find out errors
    ANI 00010000B
    JNZ SEEKERROR
```

Typical

```
    ; READ SECTOR
    MVI B, 128    ; No of data bytes
    LXI H, BUFPTR ; Buffer pointer
    MVI A, SECTORNO
    OUT SECPORT   ; sector register
    MVI A, 8CH    ; read sector command
    OUT CPORT     ; to command port
11  LOOP: IN CPORT  ; status reg.
7       ANI 00000010B ; DRQ bit
10      JZ LOOP
11      IN DPORT  ; read data
7       MOV M, A
5       INX H     ; increment pointer
4       DCR B     ; decrement counter
10      JNZ LOOP
```

∴ IT IS NOT POSSIBLE TO READ WITHOUT LOSING DATA.

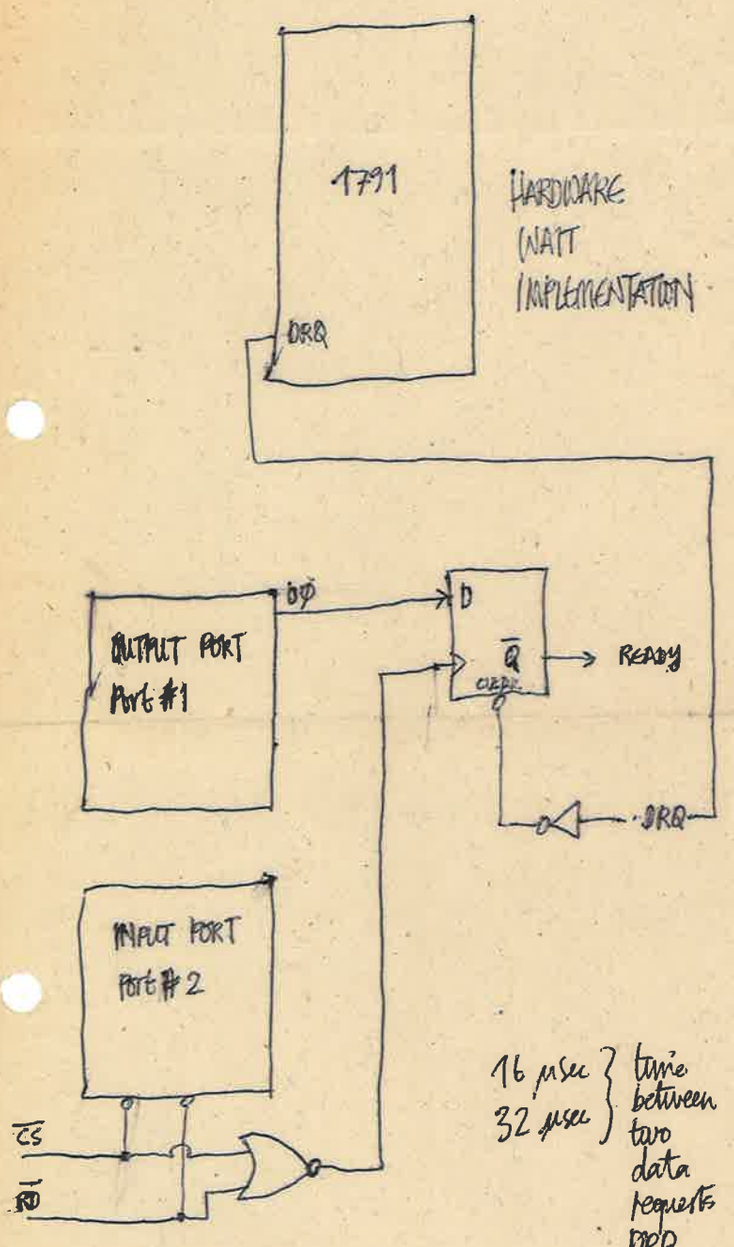
WORST CASE :

+ 52 clock cycles
7
10
11
7
10
11
7
5
4

2 MHz clock
52 μsec.
4 MHz
20.5 μsec

	READ	WRITE
8" single density	27.5 μsec	23.5 μsec.
8" double density	13.5 μsec	11.5 μsec.

Time between DRQ & INPUT instruction



16 μsec } time between two data requests DRQ
32 μsec }

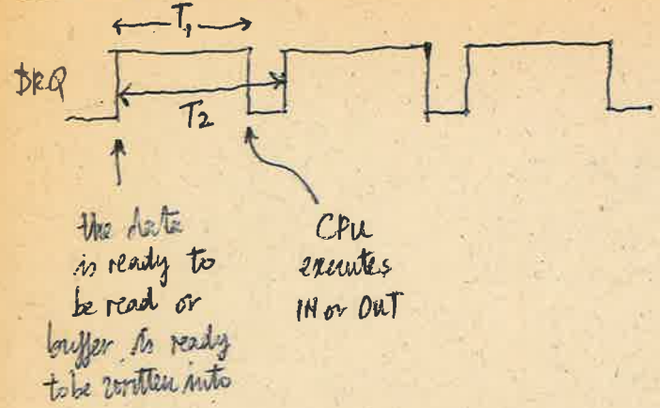
```
MVI A, 01H ; make D input high
OUT PORT1
MVI A, B4 H ; read sector
OUT CPORT
LOOP: IN PORT2 ; dummy read to put into wait
      IN DPORT
      MOV M, A
      DCR B → INX H
      JNZ LOOP
```

11
11
7
5
4
10
48

24 μsec.
12 μsec.

514 9/1/84

Floppy disk controller service time requirements



	T_1 max READ	T_1 max WRITE	T_2
8" single density	27.5 μsec	23.5 μsec	32 μsec
double density	13.5 μsec	11.5 μsec	16 μsec
5¼ single density	55 μsec	47 μsec	64 μsec
double density	27.5 μsec	23.5 μsec	32 μsec

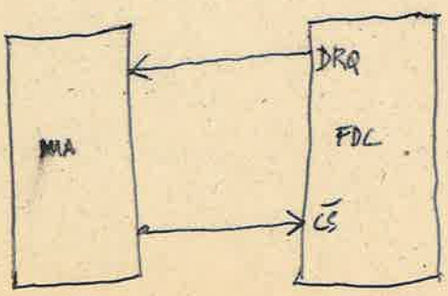
T_2 is fixed.

Continue to previous week's example:

```
BACK :  IN PORT2  ; dummy read   11
        IN DPORT                 11
        MOV M,A                   7
        INX H                     5
        DCR B                     4
        JNZ BACK                 10
                                 48
```

2 MHz Clock: 24 μsec
4 MHz 12 μsec

DMA + FDC :



CRT Controller

80 × 24 chars. 1920

Frame rate = $\frac{1}{50}$ sec.

256 lines/frame = 24 lines × 10 + Margin (16)

For each line ⇒ 80 char/line × 8 dots/char

= 640 dots/line

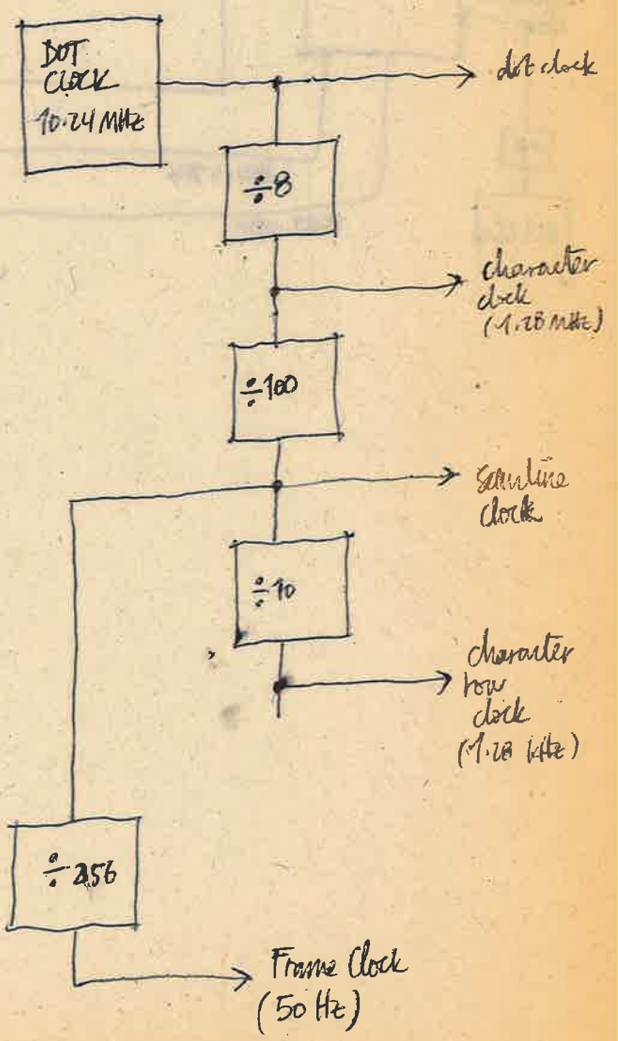
time for retrace → 160 dots/line retrace period

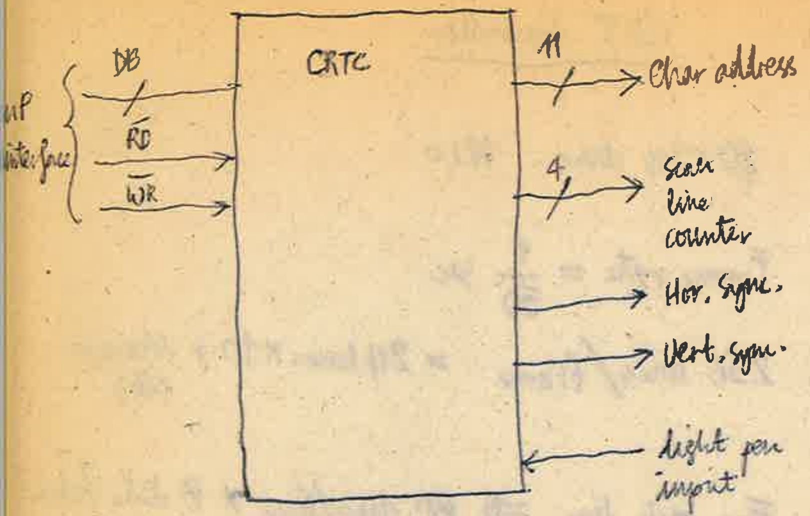
800 dots/line

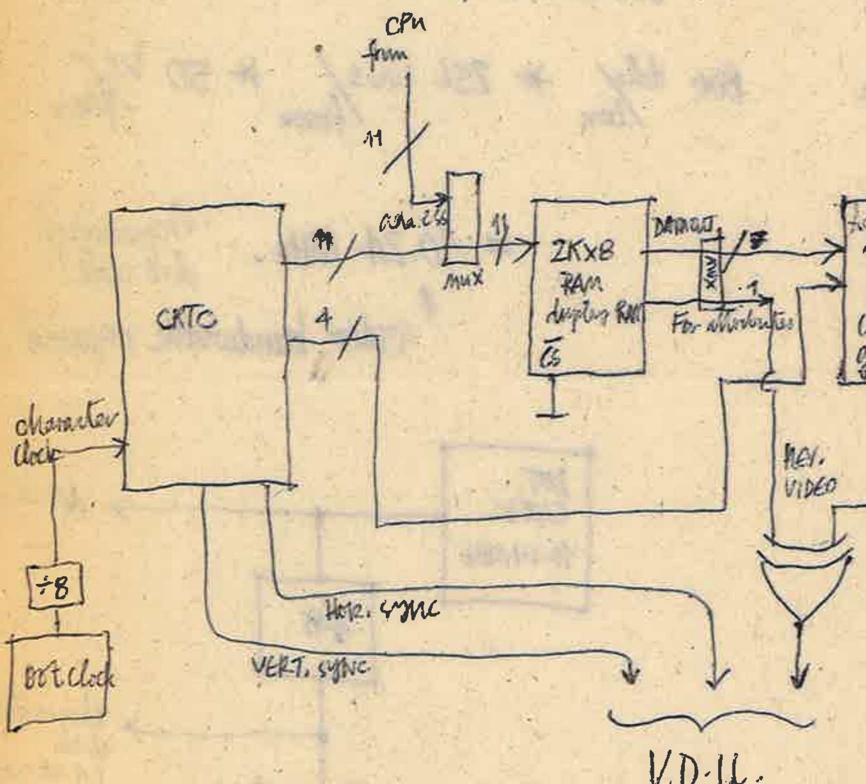
800 dots/line * 256 lines/frame * 50 sec/frame

= 10.24 MHz. ← frequency of dot clock.

↑ Video bandwidth required




CRTC
DB
RD
WR
uP interface
11
Char address
4
Scan line counter
Hor. Sync.
Vert. Sync.
light pen input


CPU
from
11
MUX
2KxB
RAM
display RAM
CS
DATA OUT
7
1
For attributes
2KxB
ROM
Character generator
8 bits
DATA OUT
DOT CLOCK
SHIFT REG
CRTC
4
character clock
÷8
DOT clock
REV. VIDEO
HOR. SYNC
VERT. SYNC
V.D.U.

541 9/1/84

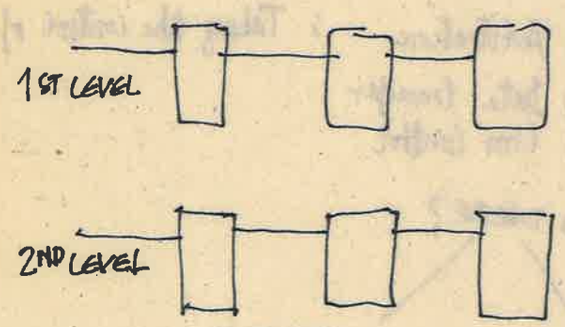
Interrupts

i) Single level, single priority

ii) Multiple " " "
→ interrupting device sends a code.

iii) Single level, multiple priority
eg: daisy chain.

iv) Multiple level, multiple priority



I/O CHANNELS (Processors)

* SELECTOR CHANNEL: they can do very high speed transfer.
eg. From/To DISKS

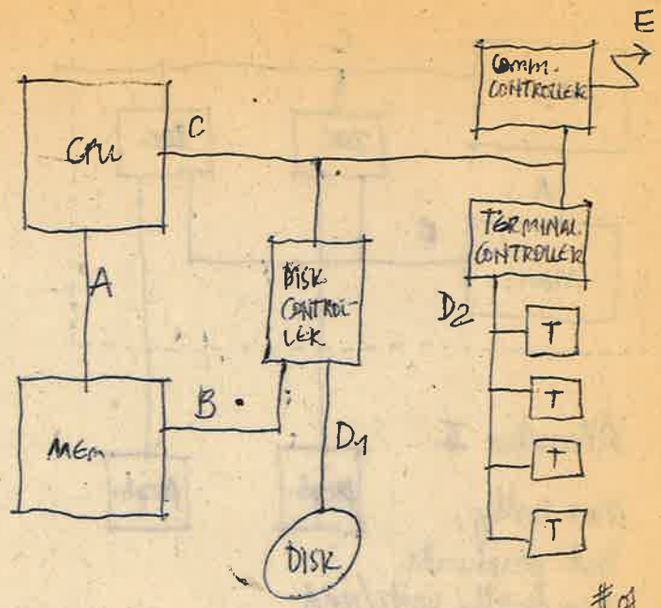
* MULTIPLEXER CHANNEL: For slower I/O devices
eg. printers

* BLOCK MULTIPLEXER CHANNEL: It multiplexes data at block level.
eg. 2 disks at a time

12/1/84

Requirements of Bus System:

i) Establish Communication Pathway
ii) Well specified and documented
iii) No unnecessary strict performance criteria.
iv) Cost of the bus compared to computer system should be justified.
v) Allow future expansions
vi) Able to manufacture & test the bus without too much "hand tuning."



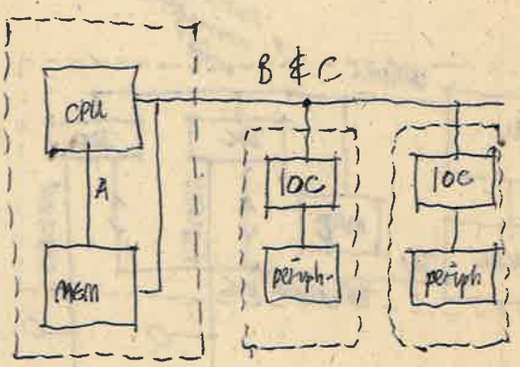
TYPE A BUS : CPU ↔ MEMORY (# of connections: small 4-16)
Data + Address
large (20 bits)

B : CONTROLLER ↔ MEMORY (width 4-16)

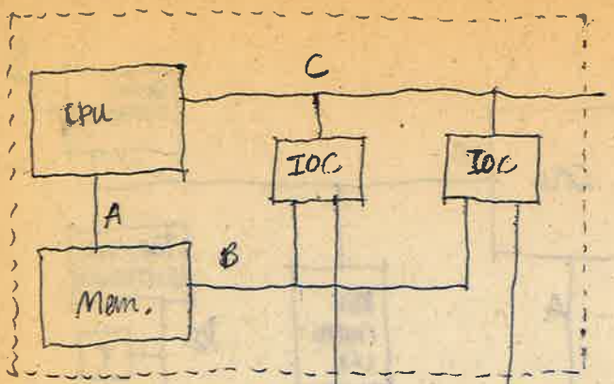
C, D, E : No need mem-addressing (medium (64) / small (16))

TYPE	LATENCY (Time needed to start transfer)	BANDWIDTH
A	LOW	LARGE
B	HIGH	LARGE
C	MEDIUM	SMALL
D	MEDIUM-HIGH	SMALL-LARGE
E	"	"

Implementation of Buses



Alternative I

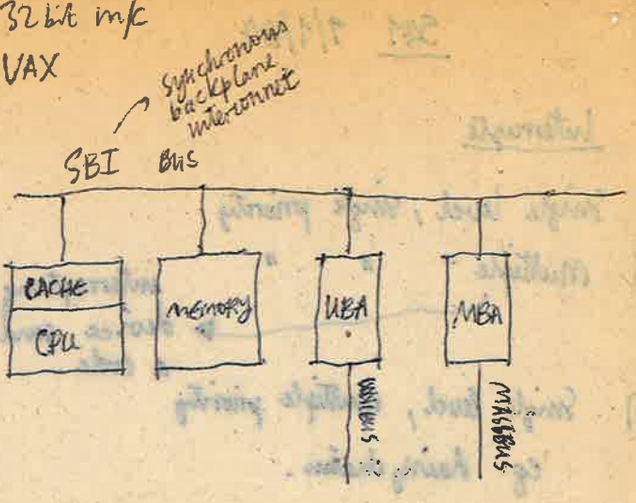


Alternative II

periph.
periph.

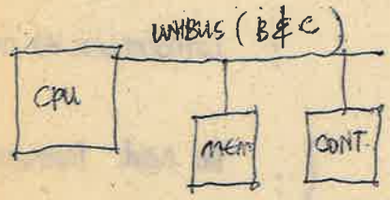
More costly;
but peripherals
may directly write/read
memory.

32 bit m/c
VAX



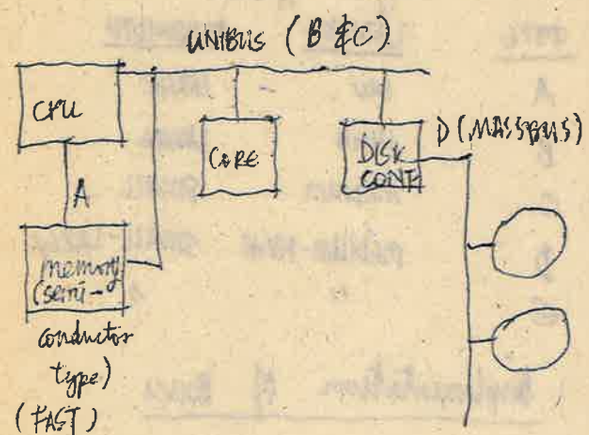
PDP-11 FAMILY

11/20

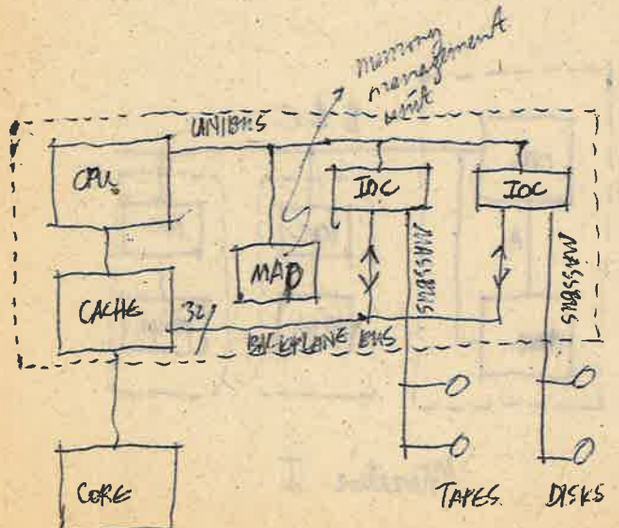


18 bit address bus

11/45



11/70



Steps to transfer data

i) Arbitration ; Taking the control of the V bus
ii) Data transfer
iii) Error control

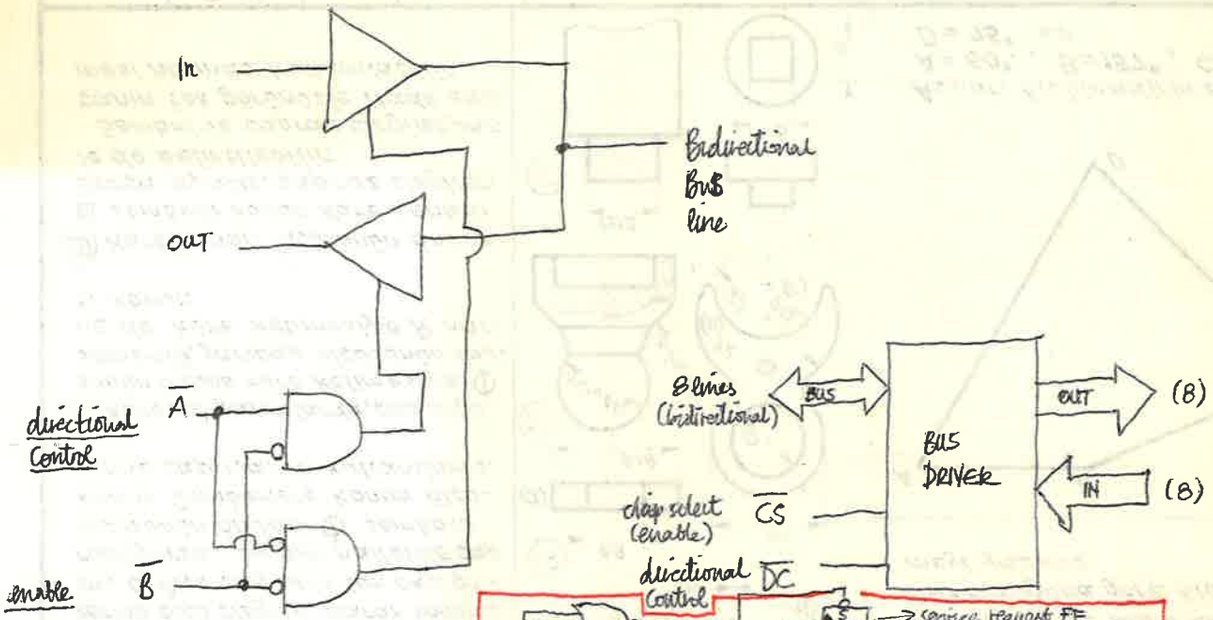
WHERE?
- CENTRALIZED
- DISTRIBUTED

HOW? — PRIORITY
- DEMOCRATIC
- SEQUENTIAL

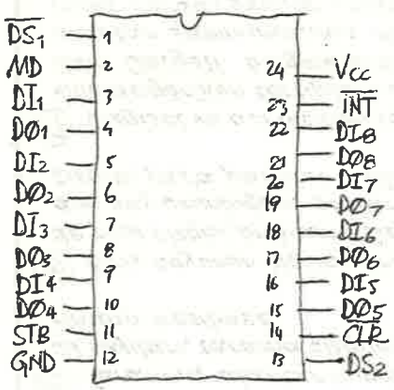
WHEN? — FIXED (Synchronous)
- VARIABLE

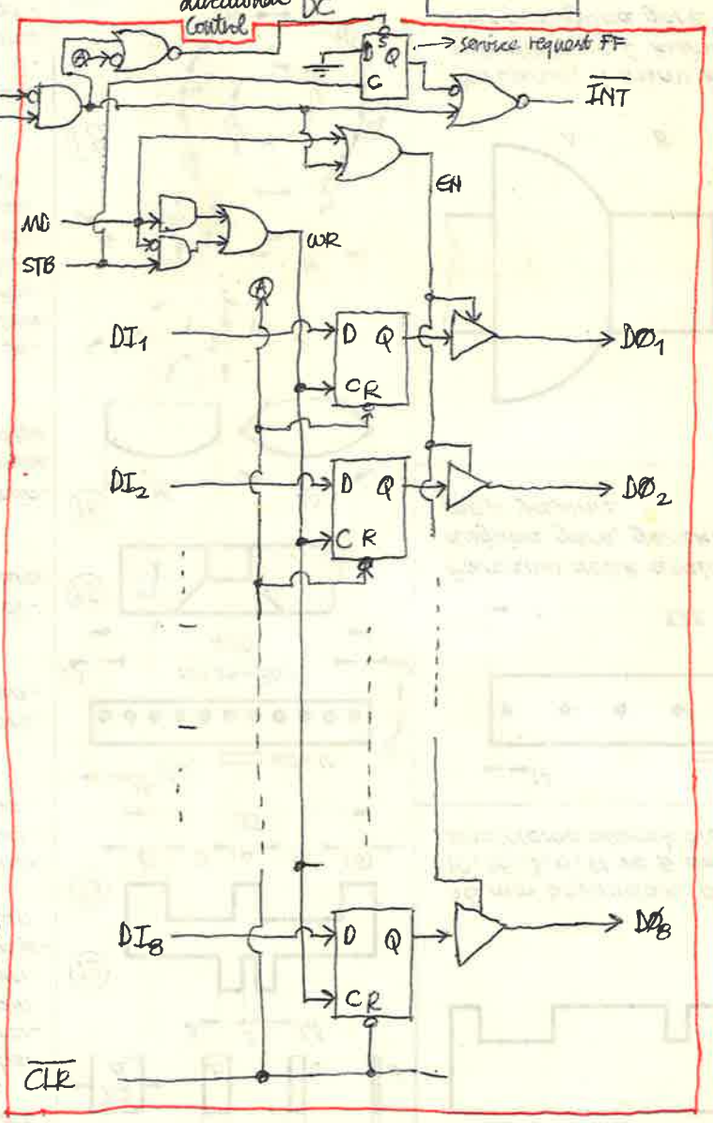
Check bits
Acknowledgements
Time out
Retry
Error logging

Bidirectional control :



8212 Port





Pin names :

$DI_1 - DI_8$: DATA IN
$DO_1 - DO_8$: DATA OUT
$\overline{DS_1}$-DS_2 : DEVICE SELECT
MD : MODE
STB : STROBE
$\overline{INT}$: INTERRUPT (Active low)
$\overline{CLR}$: CLEAR (Active low)

Data latches are level clocked. That is, as long as clock is high, Q follows D, when the clock goes low, then, at the data present at the instance of the negative going edge is latched.
$\overline{CLR}$: causes the data at all Qs go to zero. But this does

instruction types:

1. Transfer
2. Arithmetic
3. Logical
4. Shift and rotate
5. Indexing and counting
6. Bit manipulation
7. Looping
8. Branching
9. I/O communication and transfer.

addressing modes:

1. Direct addressing
2. Register addressing
3. Indirect (or deferred addressing)
4. Base addressing
5. Indexing
6. Auto incrementing / decrementing
7. Base page addressing
8. current page addressing
9. Relative (or PC) addressing
10. Immediate (actually, this is not addressing)

frequently used 8080 instructions

1 byte instructions:
- MOV r1, r2 (Move register) (r1) ← (r2)
- MOV r, M (Move from memory) (r) ← ((H)(L))
- MOV M, r (Move to memory) ((H)(L)) ← (r)
- INX rp (increment register pair) (rp) ← (rp)+1

2 bytes:
- MVI r, data (Move immediate) (r) ← (byte2)
- MVI M, data (Move to memory immediate) ((H)(L)) ← (byte2)

3 bytes:
- LXI rp, data16 (load register pair immediate) (rh) ← (byte3), (rl) ← (byte2)

op code
low-order data
high-order "

1 byte:
- ADD r (Add register) (A) ← (A)+(r)
- ADD M (Add memory) (A) ← (A) + ((H)(L))
- SUB r (Subtract register) (A) ← (A) − (r)
- SUB M (Subtract memory) (A) ← (A) − ((H)(L))

2 bytes:
- ADI data (Add immediate) (A) ← (A) + (byte2)
- SUI data (Subtract immediate) (A) ← (A) − (byte2)

3 byte:
- LDA addr. (Load accumulator direct) (A) ← ((byte3)(byte2))
- STA addr. (Store accu. direct) ((byte3)(byte2)) ← (A)

- ADC r (add register with carry) (A) ← (A) + (r) + (CY)
- ADC M (add memory with carry) (A) ← (A) + ((H)(L)) + (C
- 2 byte: ACI data (add immediate with carry) (A) ← (A) + (byte2) + (C
- SBB r (subtract register with borrow) (A) ← (A) − (r) − (CY)
- SBB M (subtract memory with borrow) (A) ← (A) − ((H)(L)) − (CY)
- 2 byte: SBI data (Subtract immediate with borrow) (A) ← (A) − (byte2) − (CY)

XCHG (Exchange H and L with D and E)
(H) ↔ (D)
(L) ↔ (E)

CMP r (Compare register) (A) − (r)
"The content of register r is subtracted from the content of accumulator. The accumulator remains unchanged. The condition flags are set as result of subtraction"

CMP M (compare memory) (A) − ((H)(L))
2 bytes: CPI data (Compare immediate) (A) − (byte2)

CMC (complement carry) (CY) ← (C̄Y)
STC (set carry) (CY) ← 1
HLT (Halt)
NOP (No operation)

increment decrement instructions:
- INR r (increment register) (r) ← (r)+1
- DCR r (decrement register) (r) ← (r)−1
- INR M (increment memory) ((H)(L)) ← ((H)(L))+1
- DCR M
- INX rp (increment register pair)
- DCR rp

3 bytes: DAD ... (add register pair to (H)(L))

Logic instructions

1 byte:
- ANA r (And register) (A) ← (A) ∧ (r)
- ANA M (And memory) (A) ← (A) ∧ ((H)(L))
- ORA r (OR register) (A) ← (A) ∨ (r)
- ORA M (OR memory) (A) ← (A) ∨ ((H)(L))

2 bytes:
- ANI data (AND immediate) (A) ← (A) ∧ (byte2)
- ORI data (OR immediate) (A) ← (A) ∨ (byte2)

1 byte:
- XRA r (exclusive OR register) (A) ← (A) ∀ (r)
- XRA M (exclusive OR memory) (A) ← (A) ∀ ((H)(L))

2 bytes:
- XRI data (exclusive OR immediate) (A) ← (A) ∀ (byte2)
- CMA (complement accumulator) (A) ← (Ā)

1 byte:
- LDAX rp (load accumulator indirect) (A) ← ((rp)) — only BC DE can be used.
- STAX rp (Store accumulator indirect) ((rp)) ← (A)

3 bytes:
- LHLD addr (Load H and L direct) (L) ← ((Byte3)(byte2)), (H) ← ((byte3)(byte2)+1)

Stack instructions :

PUSH rp. (Push)

((SP)−1) ← (rh)
((SP)−2) ← (rl)
(SP) ← (SP)−2

POP rp (Pop)

(rl) ← ((SP))
(rh) ← ((SP)+1)
(SP) ← (SP)+2

XTHL (Exchange Stack top with H and L)

(L) ← ((SP))
(H) ← ((SP)+1)

SPHL (Move HL to SP)

(SP) ← (H)(L)

PUSH PSW (Push processor status word)

((SP)−1) ← (A)
$((SP)-2)_0$ ← (CY)
$((SP)-2)_1$ ← 1
$((SP)-2)_2$ ← (P)
$((SP)-2)_3$ ← 0
$((SP)-2)_4$ ← (AC)
$((SP)-2)_5$ ← 0
$((SP)-2)_6$ ← (Z)
$((SP)-2)_7$ ← (S)
(SP) ← (SP)−2

POP PSW (Pop processor status word)

$(CY) \leftarrow ((SP))_0$
$(P) \leftarrow ((SP))_2$
$(AC) \leftarrow ((SP))_4$
$(Z) \leftarrow ((SP))_6$
$(S) \leftarrow ((SP))_7$
(A) ← ((SP)+1)
(SP) ← (SP)+2

Branching instructions of 8080

3 bytes { JMP addr (Jump) (PC) ← (byte3)(byte2)

J<condition> addr (Conditional jump)

If (CCC) ; (PC) ← (byte3)(byte2)

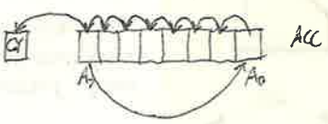
1 1	C C C	0 1 0
low order addr		
high order addr		

Condition	CCC
NZ – not zero (Z=0)	000
Z – zero (Z=1)	001
NC – no carry (CY=0)	010
C – carry (CY=1)	011
PO – parity odd (P=0)	100
PE – parity even (P=1)	101
P – plus (S=0)	110
M – minus (S=1)	111

Rotations

RLC (Rotate left) : $(A_{n+1}) \leftarrow (A_n)$; $(A_0) \leftarrow (A_7)$

$(CY) \leftarrow (A_7)$



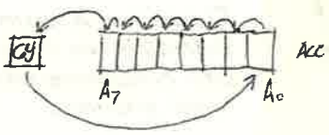
RRC (Rotate right)

$(A_n) \leftarrow (A_{n+1})$ $(A_7) \leftarrow (A_0)$

$(CY) \leftarrow (A_0)$

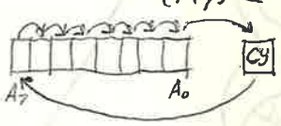
RAL (Rotate left through carry) $(A_{n+1}) \leftarrow (A_n)$ $(CY) \leftarrow (A_7)$

$(A_0) \leftarrow (CY)$

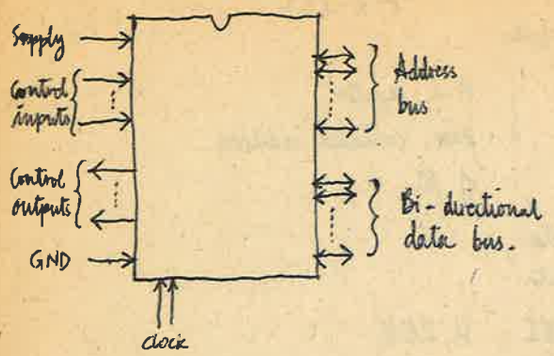


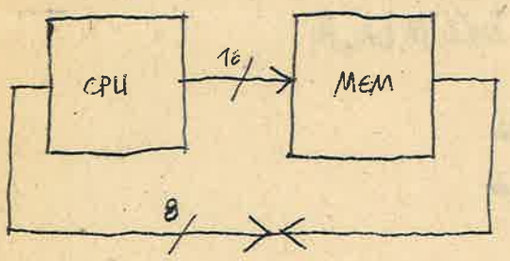
RAR (Rotate right through carry) $(A_n) \leftarrow (A_{n+1})$ $(CY) \leftarrow (A_0)$

$(A_7) \leftarrow (CY)$



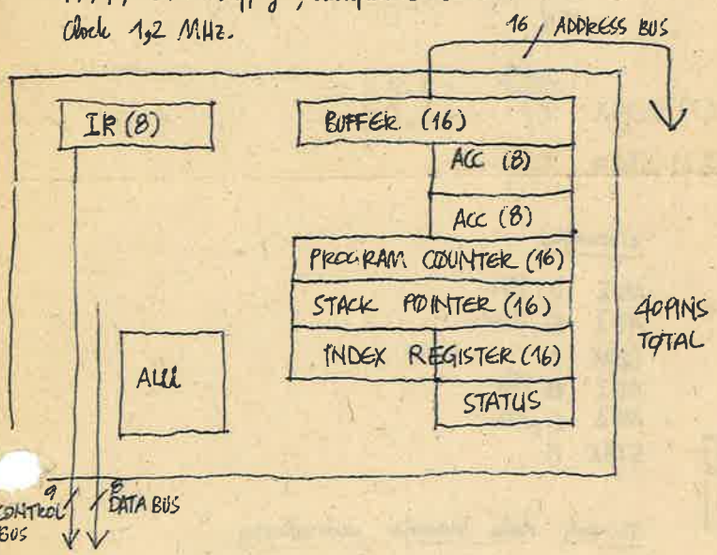
Generalized µP





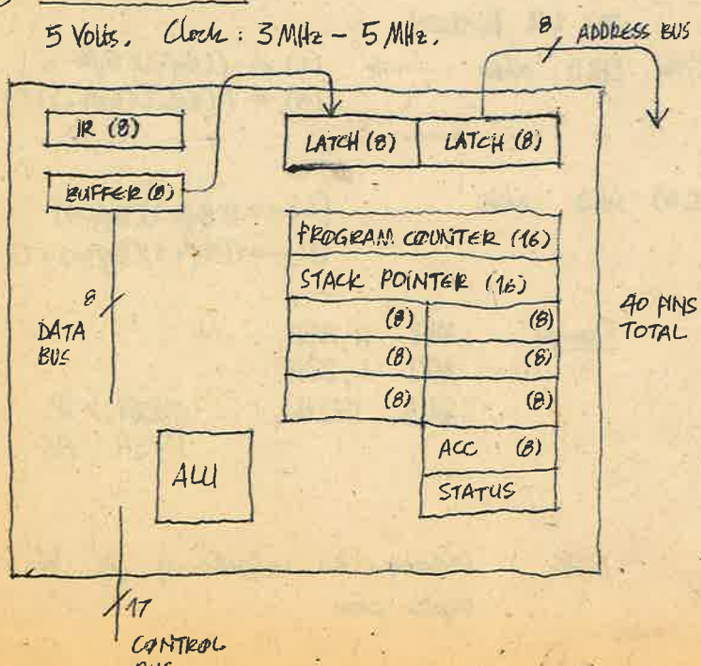
① 6800 8 bit µP, MOTOROLA

1974, 5 Volts supply, dissipation 500mW
Clock 1,2 MHz.



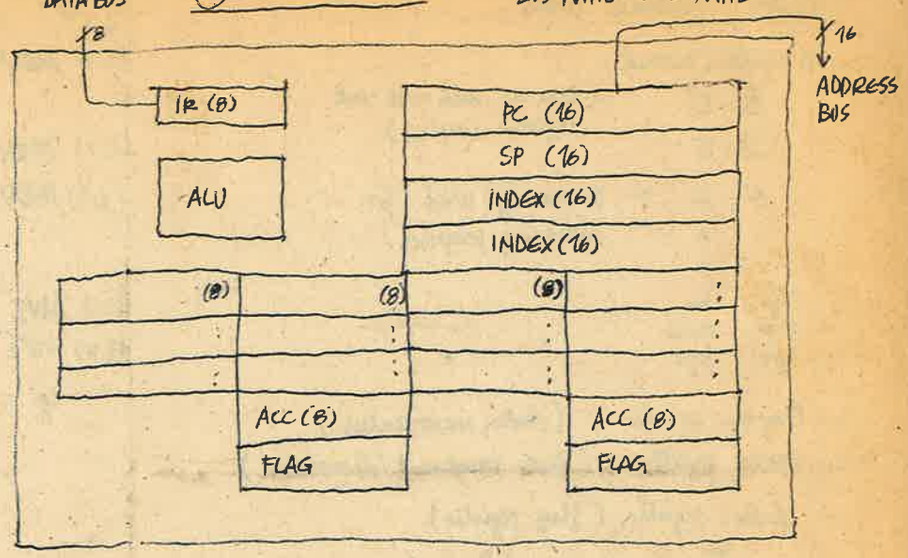
② 8085 INTEL

5 Volts. Clock: 3 MHz – 5 MHz.



③ Z80 ZILOG

CLOCK: 2.5 MHz ↔ 6 MHz



④ NSC 800 NATIONAL

CMOS processor
Internal structure: Z-80
Bus structure: 8085
≤ 100 mWatts. consumption

COURSE OUTLINE

1. Introduction to µProcessors.
2. Review of number systems.
3. Review of hardware: logic components.
4. Typical µP's: 8085
   6800
   Z80
   NSC800
5. µC system components: Memory devices and peripherals.
6. Interrupt and DMA.
7. Mass memory devices: Floppy disc drives
   Cassette drives.
8. Keyboards and CRT terminals, printers
9. Development systems, aids in debugging, emulation
10. Hardware implementation and software implementation.
11. Single chip microcomputers
12. Analog microprocessors.

BOOKS

1. Adam Osborne / Introduction to µP's.
2. 8080/8085 Assembly Language Programming.
3. MCS 80/MCS85 Family Users Manual / Intel Corp.
4. A. Lesea R. Zaks / µP interfacing techniques
5. Peatman / µC Based design.
6. L.A. Lawenthal / Introduction to µP's: Software, hardware, Programming / Prentice-Hall.

090382

8085A Functional Units :

- 6 registers named :
  - B - C (can be used as 16-bit register couples)
  - D - E
  - H - L ← Commonly used for addressing purposes.

  H : High order byte
  L : Low order byte
- Program counter : (Auto incremented)
- Stack pointer. (Auto increment / decrement)
- Status register (Flag register)

7TH					0TH
S	Z	AC	P		CY

(all of the 8-bits are not used)

CY : Carry (1 denotes overflow)

AC : Auxiliary carry ⟶ (Useful for BCD)

eg :
```
  10101100
+ 01110100
-----------
 100100010
      ↑  ↑
     AC  CY
```

S : Sign of the number stored in the accumulator.

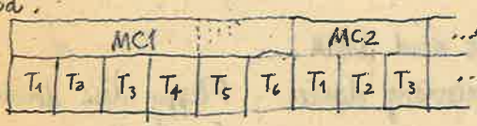
Z : Zero flag : Set if contents of A is zero.

P : Parity : Set if parity is even.

8085A uses 6.25 MHz. crystal and CLK output is equal to 6.25/2 = 3.125 MHz.

clock period : 320 nsec.

instruction fetch (OPCODE fetch) will take 4-6 period.



..MC : machine cycle.

[Execution takes 1-5 MC's]

8085A-2 : Clock frequency is higher : Period : 200 nsec.

INSTRUCTION SET OF 8085A

1 Byte instructions : There are 246 instructions.

Groups :

(1) 8 bit data transfer instructions
(2) 16 " " " "
(3) 8 " arithmetic instructions
(4) 16 " " "
(5) logic instruction group.
(6) Branch " "
(7) Stack, I/O, machine control instructions.

8-bit data transfer instructions :

(1-4) MOV r, s (r ← (s)) — r, s : registers

(2,7) MOV r, M
(2,7) MOV M, r
} H-L register pair contains address of M.

(2,7) MVI r, data
(3,10) MVI M, data

eg :
```
MVI H, 25H
MVI L, F2H
MVI M, DA
```

(4,13) LDA addr

(4,13) STA addr

Ex :
```
3A
15      } (A) ← (2515H)
25
32
15      } (A) ⟶ (2515H)
25
```

(2,7) LDAX r.p (register pair : B-C, D-E, H-L)

(2,7) STAX r.p.

Example

```
MVI D, 11H
MVI E, 22H
LDAX D
MVI B, 11H
MVI L, 23H
STAX H
```

16-bit data transfer instructions :

(3,10) LXI rp, data 16 — rp : BC, DE, HL, SP

ex : LXI D, 1122H

(5,16) LHLD addr ⟶ (L) ← ((Byte3)(Byte2))
(H) ← ((Byte3)(Byte2)+1)

(5,16) SHLD addr (L) ⟶ ((Byte3)(Byte2))
(H) ⟶ ((Byte3)(Byte2)+1)

Example :
```
MVI H, A5H
MVI L, D2H
SHLD 1122H :   1122H : D2
                1123H : A5
```

XCHG : exchanges the contents of HL, DE register pairs