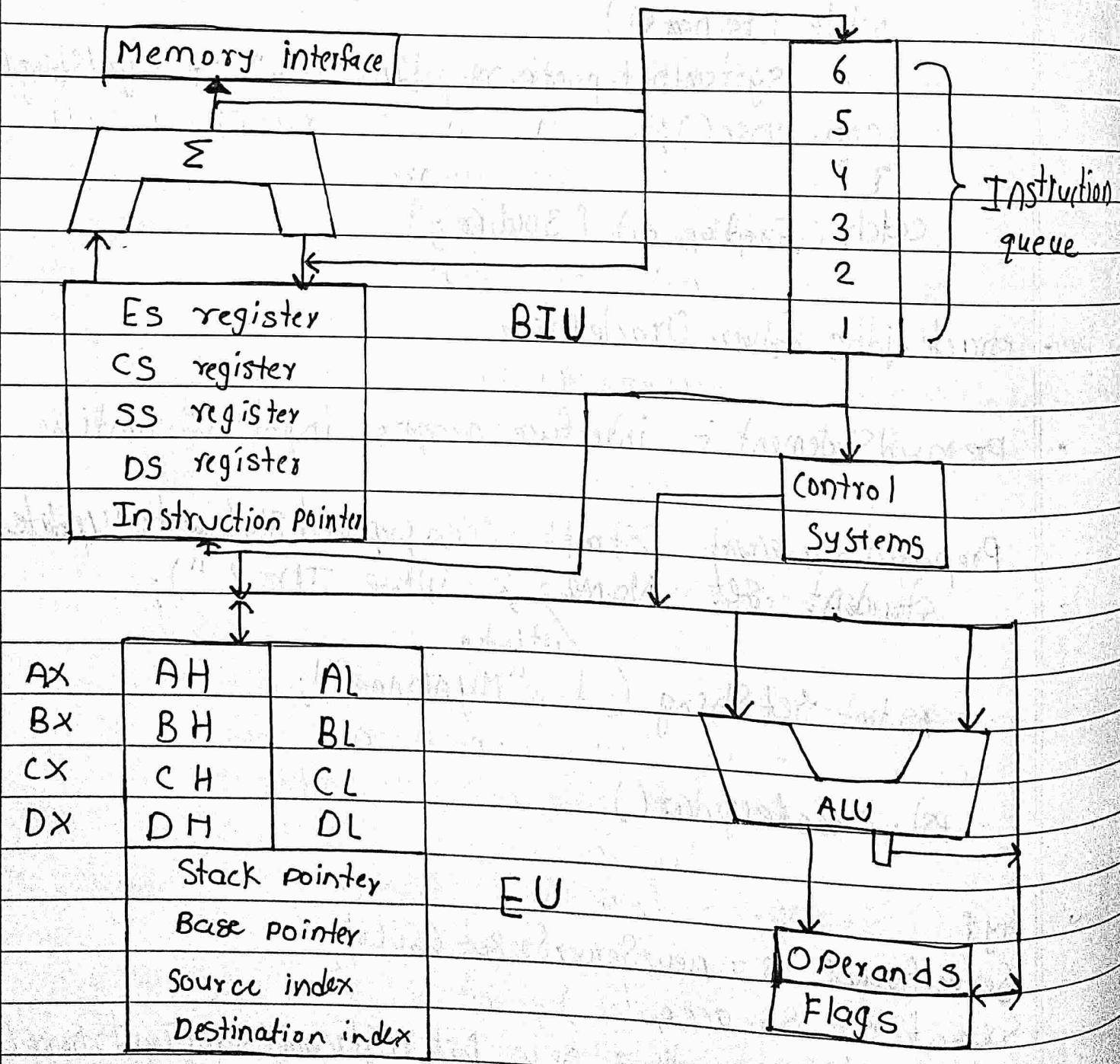
CSE2006 - Microprocessor & Interfacing
Fall semester 2022-23

Name :- Ishan Sagar Jogalekar
Reg. no :- 19BCE2250
Slot :- L43 + L44

Digital assignment-1

Q.1 Architecture of 8086 :-

Diagram :-



Memory Segmentation:

To increase execution speed & fetching speed 8086 segments the memory.

It divides into 2 units.

i) BUS interface unit (BIU) :- Bus control logic of BIU generates all the bus control signals such as read and write signals for memory and I/O.

- Handles all transfer of data & address on the buses for EU.

Function of BIU:-

- Fetch the instruction or data from memory.
- write data to memory.
- write data to port.
- Read data from port.

Contains 4 segment register.

i) Instruction pointer (IP):- Always point to next inst to be executed. Offset address is relative to CS. Always work together with CS.

ii) Segment register :-

a. CS - Points at segment containing current program
b. DS - General points at segment where variables are defined.
c. ES - Extra segment, it up to a code to define usage.
d. SS - points at segment containing Stack.

- Address generation circuit
- 6 byte pre-fetch queue

1. ii) Execution Unit :-

EU is also called a functional unit. It is part of CPU that performs the operations & calculations are instructed by computer program.

Functions :-

- To tell BIU where to fetch the inst or data from memory interface
- To decode the instructions.
- To execute the instructions.

General purpose registers :-

- AX - Accumulator register → Arithmatic logic & data transfer, Input & output
- BX - Base address register → Contain data pointer, used for data, based indexed.
- CX → Count register → Iterative code segments using loop inst. count of bits shift & rotate
- DX → Data register → used as port no. in I/O operation

ALU :-

- Stack pointer :- Always points to top item on Stack. offset address relative to SS.
- Base pointer :- Primarly used to access parameters passed via Stack.
- Source index :- Used to pointer addressing of data
- Destination index :- Can be used for pointer addressing of data.

19BCE2250- Ishan Jogalekar

1. Flag / status register :-

1. Carry flag 2. parity flag 3. Auxillary flag
4. Zero flag 5. Sign flag 6. Overflow flag

3 control flag :-

1. Trap flag 2. Interrupt flag 3. Direction flag

Q.2 Pin description of 8086 :-

GND	1	40	VCC
AD14	2	39	AD15
AD13	3	38	A16/S3
AD12	4	37	A17/S4
AD11	5	36	A18/S5
AD10	6	35	A19/S6
AD9	7	34	$\overline{BHE}$/S7
AD8	8	33	MN/$\overline{MX}$
AD7	9	32	$\overline{RD}$
AD6	10	31	$\overline{RQ}$/$\overline{GT0}$ (Hold)
AD5	11	30	$\overline{RQ}$/$\overline{GT1}$ (HLDA)
AD4	12	29	$\overline{LOCK}$ ($\overline{WR}$)
AD3	13	28	$\overline{S2}$ (M/IO)
AD2	14	27	$\overline{S1}$ (DT/$\overline{R}$)
AD1	15	26	$\overline{S0}$ ($\overline{DEN}$)
AD0	16	25	QS0 (ALE)
NMI	17	24	QS1
INTR	18	23	$\overline{TEST}$
CLK	19	22	READY
GND	20	21	RESET

2. • AD0 - AD15 (I/O) :- Address data bus :-
low order bus, AD0 - AD7 carries low order data byte while AD8 - AD15 carry higher order byte. They are multiplexed with data, for memory address.

• AD16 - A19 / $S_0 - S_1 - S_2$:-
A16 - A19 are higher order address bus. These are multiplex with status signals. In case of memory operations these pins act as address bus.

S_2	S_1	S_0	characteristics
0	0	0	Interrupt ack
0	0	1	Read I/O port
0	1	0	write I/O port
0	1	1	Halt
1	0	0	code access
1	0	1	Read memory
1	1	0	write memory
1	1	1	Passive state

• S5 → Presense of interrupts in microprocessor, serves interiput flag.

• S6 → Status of bust master for current operation, more specifically bus master.

• BHE/S7 → Bus enable Status. used to enable data onto MSH of data bus, BHE uses low signal. It is multiplex with status signal S7.

- RD(O) :- READ → Indicates processor is performing a memory or I/O read cycle, active when low.
- TEST(I) → If Test pin is low, execution continues, Otherwise processor waits in 'idle' position or state.
- INTR (Interrupt request) → level triggered input which is sampled during last clock cycle of each inst to determine the interrput ack operation.
- NMI (non-maskable interrupt) :- Edge triggred input, cause type-2 interrupt, vector look up table located in system memory.
- Reset (I) :- Cause processor to immediately terminate it present activity. Signal is high for at least last 4 clock cycles.
- Ready (I) :- Acknowldgement from addressed memory or I/O device
- CLK (I) :- clock, Provide basic time for processer & bus controller.
- MN / MX (I) :- Indicates mode on which processor is working.
- WR(O) :- Indicates processor is performing a write memory or write I/O cycle.
- INTA(O) :- Interrupt acknowledge → use as read strobe for interrupt ack cycles.

2. • ALE(O) :- Address latch enable → provided by processor to latch address.

• DT/R(O) :- Data trasmit, use do to control direct of data flow.

• DEN (O) :- Data enable, provide as Output enable for 8286/8287 in a min system.

• HOLD & HLDA (I/O) :- HOLD indicated that another master has been requesting a bus, Active high (1), it issues HLDA acknowledgment.

• QS0, QS1 (O) :- These signals indicate the status of internal 8086.

QS0	QS1	Status
0	0	No operation
1	0	First byte of OP code from queue
0	1	Empty queue
1	1	Subsequent byte from queue

• RQ/GT0 & RQ/GT1 (I/O) :- Request/Grant → Pins are used by other processor in multi processors orginization.

19BCE2250 - Ishan Jogalekar

Q.3 Addressing modes :-

The way in which the processor gets data from users is called addressing modes.
It can get data from different sources,
- Register - By instructions

i) Implied mode :- operand specified in inst. 8 or 16 bit data is part of inst. Zero AI.
Eg. CLC

ii) Immediate Mode :- Data placed in address field of inst one AI.
Eg: MOV AL, 35H
Limitation → Range of data is limited by size of address field.

iii) Register mode :- operand is placed in 8 or 16 bit register, it is registered specific mode.
Eg. MOV AX, CX

iv) Register indirect mode :- Data will be placed in memory location, address of memory will place in register. Register will be part of instruction.

Eg: MOV AX, [BX]

v) Auto index or increment mode :-
- Address of operand is placed in register specified in instruction.
- After accessing operand, address in register is incremented to point to next memory location
Eg. ADD R1, (R2)+

3. vi) Direct addressing :- operands address is given in inst an 8 bit or 16 bit element.

eg: ADD AL, [0301]

vii) Indirect addressing :- Address field contains the address of EA, requires 2 references a. Register indirect b. Memory indirect.

viii) Indexed Addressing mode :- Operands address is sum of content in index register SI or DI and 8 bit or 16 bit displacement.

Eg: MOV AX [SI+05]

ix) Based addressing mode :- EA is obtained by adding base register value to indexed register.

eg: ADD AX, [SI+BX]

3. Instruction set of 8086 microprocessor :-

- Data transfer :-

1) MOV :- Copy the byte or word from source to distination
2) PPUSH :- Used to put word at top of stack.
3) POP :- Used to get word from top of stack to the provided location.
4) POPA :- used to get works from stack to all registers.
5) XCHG :- Used to exchange the data from two location
6) XLAT :- Used to traskate byte in AL using table.
7) IN :- Used to read byte or word from provided register from memory.
8) Out :- used to send out byte or word from Ax.
9) LEA :- used to load the address of operand into provided register.
10) LDS :- Used to load DS register and other provided register from memory.
11) LES :- used to load LES register & other.
12) LAHF :- Used to store AH register to low byte of flag.
13) SAHF :- used to load AH register to flag.
14) PUSHF :- Copy of flag register at top of stack.
15) POPF :- Copy word at top of stack.

- Arithmetic instrictions :-

1) ADD - Used to add given byte or word.
2) ADC - used add with carry.
3) INC - increment provided by 1.
4) AAA - Adjust ASCII after addition
5) DAA - Adjust decimal after add/sub op.
6) SUB - Sub byte from byte/word.

3. 7) SBB - used to do substraction with borrow.
8) DEC - Decrement by 1.
9) NPG - negate by each bit 1 / 2'S complement.
10) CMP - compare 2 provided byte/ word.
11) DAS - Adjust decimal after SUB.
12) AAS - Adjust ASCII Code after SUB.
13) MUL - multiply unsigned byte by byte / word by word.
14) IMUL - multiply signed byte by byte / word by word.
15) AAM - Adjust ASSCII code after Mul.
16) DIV - Divide unsigned word by byte.
17) AAD - Adjust ASCII code after div.
18) IDIV - Divide signed word by byte
19) CBW - Fill upper byte of word with copies of sign bit of lower byte
20) CWD - Fill upper word of double word.

- Bit manipulation instructions :-

1) NOT - Invert each bit of byte or word.
2) AND - adding each bit in byte / word.
3) OR - used to multiplying each bit in byte / word.
4) XOR - Perfor Ex-OR Operation over each bit.
5) Test - Add operands to update flags
6) SHL / SAL - shift bits of byte / words towards left. put zero in LSB.
7) SHR - shift bits of byte / words towards right. Put zero is MSB.
8) SAR - shift bits of byte towards right and copy ~~new~~ old MSB into new MSB.
9) ROL - rotate bits towards left & to CF.
10) ROR - rotate bits towards right & to CF.
11) RCR - rotate bits towards right LSB → CF → MSB.

3. 12) RCL - rotate bits towards left

String instructions:-

1) REP - Repeat inst. till CX ≠ 0.
2) REPE/ REPZ - repeat inst. till CX ≠ 0 / ZF = 1.
3) REPNE/REPNZ - repeat inst. till CX = 0 / ZF = 1.
4) MOVS/ B/W - Move one string to another.
5) COMS/COMSB - compare two string bytes / words.
6) NS/NSB/ NNSW - Input string from I/O part to memory
7) OUTS/OUTSB/OUTSW - output string from I/O part.
8) SCAS/ SCASB - Scan a string & compare its byte.
9) LODS/ LODSB - Store string byte into AL.

Program execution transfer instructions:-

1) CALL - call procedure & save their return address to register
2) RET - return from procesdure to main program
3) JMP - Jump to provided address
4) JA
5) JAE
6) JC - Jump if carry flag = 1
7) JE/JZ - Jump if ZF = 1
8) JG/JNLE 9) JGE/JNL 10) JL/ JNGE 11) JEE/JNG
12) JNC - Jump if no carry flag
13) JNE/JNZ - jump if ZF = 0
14) JNO - jump if over flow flag, OF = 0.
15) JPO - jump if parity flag, PF = 0
16) JNS - jump if not sign SF = 0
17) JO - jum if overflow flag, OF = 1

3. 18) JP / JPE - Jump if PF = 1
19) JS - Jump if Sign flag, SF = 1

Processor control instruction :-

1) STC - set carry flag CF = 1
2) CLC = Clear / rest CF to 0.
3) CMC - Post complement at state of CF.
4) STD - set direction flag, DF = 1
5) CLD - Clear / rest DF to 0.
6) STI - set interrupt flag, IF = 1.
7) CLI - Clear / rest IF to 0.

Iteration control instruction :-

1) LOOP - Loop group of instruction until condition satisfies i.e CX = 0
2) LOOPE / LOOPZ
3) LOOPNE / LOOPNZ - Group of inst till satisfies ZF = 0 & CF = 0

Interrupt instruction :-

1) INT 2) INTO 3) IRET