Microprocessor Basics

1. What are the differences between microprocessors and microcontrollers?
   ⇒ slide
2. For each of the following devices, would you use a microcontroller or a microprocessor?
   a. Keyboard - Microcontroller
   b. Mouse - Microcontroller
   c. Headphone - Microcontroller
   d. Computer - Microprocessor
   e. TV Remote - Microcontroller
   f. Smart TV - Microprocessor
   g. Regular fridge - MIcrocontroller
   h. Mobile phone - Microprocessor

3. Briefly describe the 3 steps involved in executing instructions inside a microprocessor
   ⇒ Fetch → Decode → Execute (Look up the description in slides)
4. What are the differences between RAM and ROM?
   ⇒
5. What are the 3 types of bus inside a microprocessor? Briefly describe the purpose of each bus.
   ⇒ Address, Data, Control (Look up the description in slides)
6. ATMega328p is an 8 bit microcontroller. It has 32KB on-board flash memory (ROM)
   a. What is the data bus size of the microcontroller?
      ⇒ 8 bit microcontroller → 8 bit data bus (Word size = 8 bit)
   b. What is the minimum address bus required for accessing the on-board flash memory?
      ⇒ 32KB flash memory = 2^{15} bytes → minimum 15 bit address bus required
7. Calculate the maximum supported RAM size for the following address bus lengths:
   a. 16 bit bus → 64 KBytes memory(10 bit supports 1KByte)
   b. 20 bit bus → 1 MBytes memory (20 bit supports 1MByte)
   c. 24 bit bus → 16 MBytes memory
   d. 32 bit bus → 4 GBytes memory (30 bit supports 1GByte)
   e. 64 bit bus → 2^{64} Bytes memory

Microprocessor Architecture

1. Which functional unit does the following register belong to?
    a. Segment Register ⇒ BIU
    b. Instruction Pointer ⇒ BIU
    c. Index Register ⇒ EU
    d. General Purpose Register ⇒ EU
    e. Instruction Queue ⇒ BIU
    f. Flags ⇒ EU
2. What is the advantage of an instruction queue in a processor?
3. With an example, show how instruction queuing can speed up the time required for processes in the 8086 microprocessor.
4. What is the purpose of segment registers in the 8086 microprocessor?
5. What is the role of flag register in the 8086 microprocessor

Memory Partitioning and Segmentation

1. What is the bit length of the internal registers of the 8086 processor?
   ⇒ 16 bit
2. Why cannot the 8086 processor access the full memory without memory partitioning?
   ⇒ Address bus is 20 bit but registers are 16 bit
3. What do you understand by the term "Memory Segmentation"?
4. What are the advantages of overlapping segmentation compared to non-overlapping segmentation?
5. In the 8086 processor, what is the minimum and maximum segment size ⇒ Minimum segment size = 10h Byte = 16 Byte
   Maximum segment size = 10000h Byte = 64 KByte

6. Find the largest and smallest possible segment address for each of the following physical addresses. Also, mention the logical addresses for each case. (Maximum segment size can be 64KBytes)
   ⇒ To find the smallest possible segment address, we assume the largest possible offset and
   To find the largest possible segment address, we assume the smallest possible offset.
   a. *12345h*
      ⇒ 12345h rounded **down to nearest 10s**= 12340h,
      largest segment address = 1234h
      12345h - FFFFh = 02346h, rounded **up to nearest 10s**= 02350h,
      Smallest segment address = 0235h
   b. *10h*
      ⇒ 00010h rounded **down to nearest 10s**= 00010h,
      largest segment address = 0001h
      00010h - FFFFh = F0011h, rounded **up to nearest 10s**= F0020h,
      Smallest segment address = F002h

7. A memory location has a physical address of 9A7B1 H. Determine the offset address if the segment number is 40FF H.
   ⇒ 9A7B1h - 40FF0h = 597C1h Offset cannot be larger than 4 hex digits (invalid address)

8. Find the smallest and the second largest segment address for each of the following physical addresses. Also, mention the logical addresses (**segment: offset pair**) for each case:
   a. FFFEFh
      ⇒ FFFEFh rounded **down to nearest 10s**= FFFE0h,
      largest segment address = FFFEh
      Second largest segment = FFFDh
      FFFEFh - FFFFh = EFFF0h, rounded **up to nearest 10s**= EFFFh,
      Smallest segment address = EFFFh
   b. 2h
      ⇒ 00002h rounded **down to nearest 10s** = 00000h
      Largest segment address = 0000h

Second largest segment = FFFFh

00002h - FFFFh = F0003h, rounded **up to nearest 10s**= F0010h,

Smallest segment address = F001h

9. Explain to which addressing mode does the instruction “RET AX” belong

   ⇒ Invalid instruction. RET does not take any operands

10.

Address	10600h	10601h	20600h	20601h	30600h	30601h
Data	12h	34h	56h	78h	10h	20h

Given DS = 2000h, SS = 1000h, CS = 3000h, BP = 0400h, DI = 0200h. Now **deduce** what data will be stored in BX if the instruction MOV BX, [BP + SI] is executed.

⇒ PA = 1000 x 10h + (0400 + 0200)h = 10600h [SS will be used for BP].

So BX will store 3412h

11. Instead of 64KB, assume the size of each segment for an 8086 is 256 bytes. In that case, deduce the 2nd last address of a segment whose starting address is 10000h.

⇒ 2^8 = 256. So the maximum 8-bit offset is FF.

2nd last address = 10000 + 00FE = 100FEh

Flag Register

1. Suppose two hexadecimal numbers (i) **96C9** and (ii) **99XY** are to be added by an Intel 8086 microprocessor. Here X and Y represent two unknown hexadecimal digits.
   a. After the 8086 **adds** the numbers (i) and (ii), deduce the minimum values of X and Y needed to get the value of AF = 1

      ⇒ 96C9 = 1001 0110 1100 1001

      99XY = 1001 1001 X Y

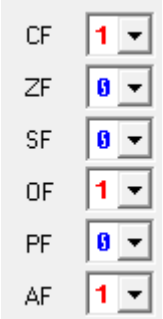
      AF = 1 is the carry over from the last four bits, so we can safely set X = 0

      To get a carry of 1, values of Y can be greater than 0111,

      Thus, minimum values of X and Y are 0000, 0111 respectively

   b. Using the deduced values of X and Y obtained from (a), find the values of PF, SF, OF and CF for the given **addition** operation.

      ⇒ The numbers to be added are 96C9, 9907. Flags will be:

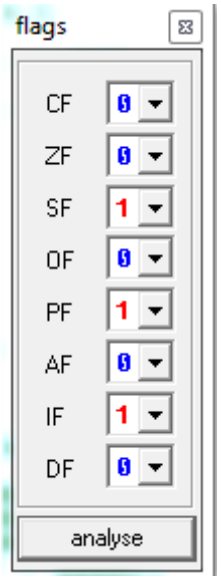


2. Calculate the OF, PF, AF, SF, and CF after the execution of the given instructions.

   >>> MOV AX, AB56h

   >>> MOV BX, 3965h

   >>> ADD AX, BX



Addressing Modes and Machine Code

1. Suppose you are given the following machine code for an instruction: **88160080h.** Deduce the original assembly language instruction denoted by the above-mentioned machine code.
   ⇒
   First 4 hex digits 8816 = 1000 1000 0001 0110

OPCODE						D	W	MOD		REG			R/M		
1	0	0	0	1	0	0	0	0	0	0	1	0	1	1	0

   D = 0, from register
   W = 0, byte size data
   MOD = 00, R/M = 110 ⇒ d16 (direct address of 16 bits)
   W = 0, REG = 010 ⇒ DL

   Last 4 hex digits 0080 ⇒ lower byte = 00, higher byte = 80

   Thus, the instruction will be **MOV [8000h], DL**

2. Consider the instruction >>> ***MOV C5A4h, CX***
   Deduce the corresponding machine code
   ⇒ Incorrect instruction, cannot move register data to immediate data.
   Correct instruction, **MOV [C5A4h], CX**
3. Convert **89806910h** from machine language to its corresponding assembly language. Show all of your work.
   ⇒
   First 4 hex digits 8980 = 1000 1001 1000 0000

OPCODE						D	W	MOD		REG			R/M		
1	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0

   D = 0, from register
   W = 1, word size data
   MOD = 10, R/M = 000 ⇒ [BX] + [SI] + d16
   W = 1, REG = 000 ⇒ AX

   Last 4 hex digits 6910 ⇒ lower byte = 69, higher byte = 10

   Thus, the instruction will be **MOV [BX+SI+1069h], AX**

4. Suppose the instruction **MOV DI, [BP+42h]**, appears in a program. What is its machine language for the given instruction?

   ⇒ 8B 7E 42

OPCODE						D	W	MOD		REG			R/M		
1	0	0	0	1	0	1	1	0	1	1	1	1	1	1	0

6. For each of the following statements, state the addressing modes.

(i) MOV BX, 1000H ⇒ Immediate

(ii) MOV DI, 1008H ⇒ Immediate

(iii) MOV AX, B [BX+SI+2]; B is an array ⇒ Base relative plus index

(iv) MOV AX, B [BX] ⇒ Register relative

7.

Address	10600h	10601h	20600h	20601h	30600h	30601h
Data	12h	34h	56h	78h	10h	20h

Given DS = 1000h, SS = 2000h, CS = 3000h, BP = 0500h, SI = 0100h. Now **deduce** what data will be stored in BX if the instruction MOV BX, [BP + SI] is executed.

⇒ PA = 2000 x 10h + (0500 + 0100)h = 20600h [SS will be used for BP],

So BX will store 7856h

8. Explain with reasoning CALL [BX] falls under which addressing mode.

⇒ Addressing Program Codes with Relative Registers (No proper name is given in the slide).

Memory Banking

1. An 8086 cpu has a RAM of 1MB split into two equal memory banks. The A_0 and $\overline{BHE}$ pins of the cpu are used to access the two memory banks. For each of the given instructions, determine the following information:

(i) Size of the data being transferred (byte/word)
(ii) Which portion of the data bus was used to transfer the data from source location to destination register
(iii) Value of A_0 and $\overline{BHE}$ required to make the transfer
(iv) Memory bank from which the data is fetched
(v) Total number of data cycles used to move data

Assuming DS=0000h, consider the following instructions-

a. MOV AH, [8086h]
(i) Byte size
(ii) D7-D0
(iii) $A_0 = 0$, $\overline{BHE} = 1$
(iv) Even bank
(v) Single cycle

b. MOV AL, [8086h] Same as above

c. MOV AX, [8086h]
(i) Word size
(ii) D15-D0
(iii) $A_0 = 0$, $\overline{BHE} = 0$
(iv) Both memory banks
(v) Single cycle

d. MOV BH, [8085h]
(i) Byte size
(ii) D15-D8
(iii) $A_0 = 1$, $\overline{BHE} = 0$
(iv) Odd bank
(v) Single cycle

e. MOV BL, [8085h] Same as above

f. MOV BX, [8085h]
(i) Word size
(ii) D15-D0
(iii) 1st bus cycle: $A_0 = 1$, $\overline{BHE} = 0$, 2nd bus cycle: $A_0 = 0$, $\overline{BHE} = 1$
(iv) Odd bank first, even bank next.
(v) Two cycle

2. Why is memory banking done in the 8086 system? What would happen if memory banking is not implemented?
⇒ To access word-sized data in a single bus cycle. Without banking, word-sized data movement would require twice as much time.

3. What do you understand by aligned word and unaligned word? Explain with an example.
⇒ Aligned word: When the higher byte is contained in the odd bank and lower byte is contained in the even bank for a word sized move operation.
Unaligned word: When the higher byte is in the even bank and the lower byte is in the odd bank for a word sized move operation

Pin Specification and Timing Diagram

1. Suppose you have an Intel 8086 which is operating at a Duty Cycle of 60% and for each clock pulse assume T_{off} = 40ns. The 8086 is now going to execute the instruction MOV [1235h], AX. Based on this, answer the following questions:

a. Estimate the frequency at which the 8086 is operating.
   ⇒ Duty Cycle = 60% = percent time of T_{on}
   Percent time of T_{off} = 40% = 40ns
   Cycle duration 100% = 100ns
   Frequency = 1/100ns = 10^7 Hz = 10MHz
b. Calculate the total time for one Instruction Cycle of the given instruction.
   ⇒ The given instruction is move operation of an unaligned word sized data
   2 bus cycles required to move the data
   Clock cycles required = 2x4 = 8
   Total time required = 8x100ns = 800ns
c. Calculate the values of the A_0 and BHE' pins during the execution of the given instruction.
   ⇒ On the first bus cycle, Odd bank is accessed. A_0 = 1 and BHE' = 0
   On the second bus cycle, even bank is accessed. A_0 = 0 and BHE' = 1

2. Suppose an 8086 is operating in a way such that T_{ON} is 1/4th of the total time required for one clock pulse. Consider T_{ON} is 30ns. Now the 8086 is going to execute the instruction MOV AX, [2315h] i.e. 16 bits of data will be read from memory.

a. Calculate the frequency in MHz at which the 8086 is operating.
   ⇒ T_{on}= ¼ of full cycle = 25% duty cycle = 30ns
   Full cycle duration = 30*4 = 120ns
   Clock frequency = 1/120ns = 8.33MHz
b. Calculate the time required for 1 Machine / Bus Cycle.
   ⇒ Time required for 1 machine cycle = 4 clock cycle = 120*4 = 480ns
c. Deduce the total time required to execute the given instruction MOV AX, [2315h].
   ⇒ Unaligned data ⟶ Two bus cycles required = 480x2 = 960ns
d. Explain with proper reasoning the values of pins A_0 and BHE' during the execution of the given instruction MOV AX, [2315h].
   ⇒ On the first bus cycle, Odd bank is accessed. A_0 = 1 and BHE' = 0
   On the second bus cycle, even bank is accessed. A_0 = 0 and BHE' = 1

4. For the following instructions, mention the states of $\overline{RD}$, $\overline{WR}$, $M/\overline{IO}$, $\overline{BHE}$ pins during the T2 cycle.

a. MOV AL, [34h]

$\overline{RD}$	$\overline{WR}$	$M/\overline{IO}$	$\overline{BHE}$
0 (read from memory)	1 (not writing)	1 (access memory)	1 (even address, byte)

b. MOV [33h], BL

$\overline{RD}$	$\overline{WR}$	$M/\overline{IO}$	$\overline{BHE}$
1 (not reading)	0 (write to memory)	1 (access memory)	0 (odd address, byte)

c. OUT 82h, AL

$\overline{RD}$	$\overline{WR}$	$M/\overline{IO}$	$\overline{BHE}$
1 (not reading)	0 (write to output)	0 (access IO)	1 (even address)

d. IN AL, 82h

$\overline{RD}$	$\overline{WR}$	$M/\overline{IO}$	$\overline{BHE}$
1 (reading input)	1 (not writing)	0 (access IO)	1 (even address)

Interrupts

1. Assume the table is a portion of the current memory address space of an Intel 8086. The microprocessor currently has the following values in its registers: SS = 2000h, SP = 1124h, CS = 3000h, IP = 1450h. Now a signal arrives at the INTR pin of this 8086.

Addr	00117h	00116h	00115h	00114h	00276h	00277h	00248h	00279h
Data	45h	86h	22h	14h	12h	34h	56h	78h

a. If the 8086 decides to service the interrupt, then do the bits of the flag register change?
⇒ Only TF and IF can change (cleared). The rest of the values will remain the same and will be popped after the interrupt service routine finishes execution.

b. If the signal is of Interrupt type 69, then deduce the values of CS and IP as the 8086 starts the service routine.
⇒ Type 69 interrupt: Interrupt vector starts at = 69x4 = 276 = 00114h
Thus, new value of IP after interrupt = (Low byte at 0114h, High byte at 0115h) = 2214h
New value of CS after interrupt = (low byte at 0116h, high byte at 0117h) = 4589h

2.

Address	0006A	0006B	0006C	0006D	0006Eh	0006F
Data	00h	08h	00h	03h	00h	05h

a. Use the given table to calculate the starting address of the ISR for interrupt type 27.
⇒ type 27 has interrupt vector at 27x4 = 108 = 6Ch
Thus, IP = 0300h, CS = 0500h.
Starting address for ISR = CS:IP = 0050h:0030h = 05300h

3.

a. A student designed a new PIC called 'PIC 2.0' which takes 20 interrupt requests via its IRR0 – IRR19 pins. The size of its Interrupt Mask register (IMR), Interrupts Service Register (ISR), and Interrupt Request Register (IRR) is 20 bits. It also has 4 Cascading (CAS) pins.
   i. Assume the value of ISR0 - ISR4 is 10000, the value of IMR0 - IMR4 is 10101, and the value of IRR0 - IRR4 is 01110. Now, explain the order in which the interrupts IR0-IR4 will be serviced. Assume IR0 has the highest priority and IR19 has the least priority.
   ⇒ IMR0, IMR2, IMR4 are masked, they will be disabled.
   IRR1, IRR2, IRR3 have received interrupt signals, among which IRR2 is disabled. Between the remaining two, IRR1 will be serviced before IRR3
   ISR0 = 1, meaning IRR0 is already being serviced.
   Thus, the servicing order will be IRR0⟶ IRR1⟶IRR3

4. Assume the CS value required to locate an ISR is CBCCh and is stored at memory locations 000AEh and 000AFh respectively. Additionally assume the Interrupt vector (IV) of that ISR is CCEF2h. Hence, deduce the Interrupt Type that caused the aforementioned ISR and the value that will be stored at 000ADh.

⇒ IP = CCEF2 - (CBCCx10) = 1232; 000AD will store 12h

000AC h = 172 d; So interrupt type = 172/4 = Type 43

1. Explain what is variable addressing? Which register is used for variable addressing?

⇒ Variable addressing uses the DX register to store the address of input/output devices. The formats used for variable addressing are:

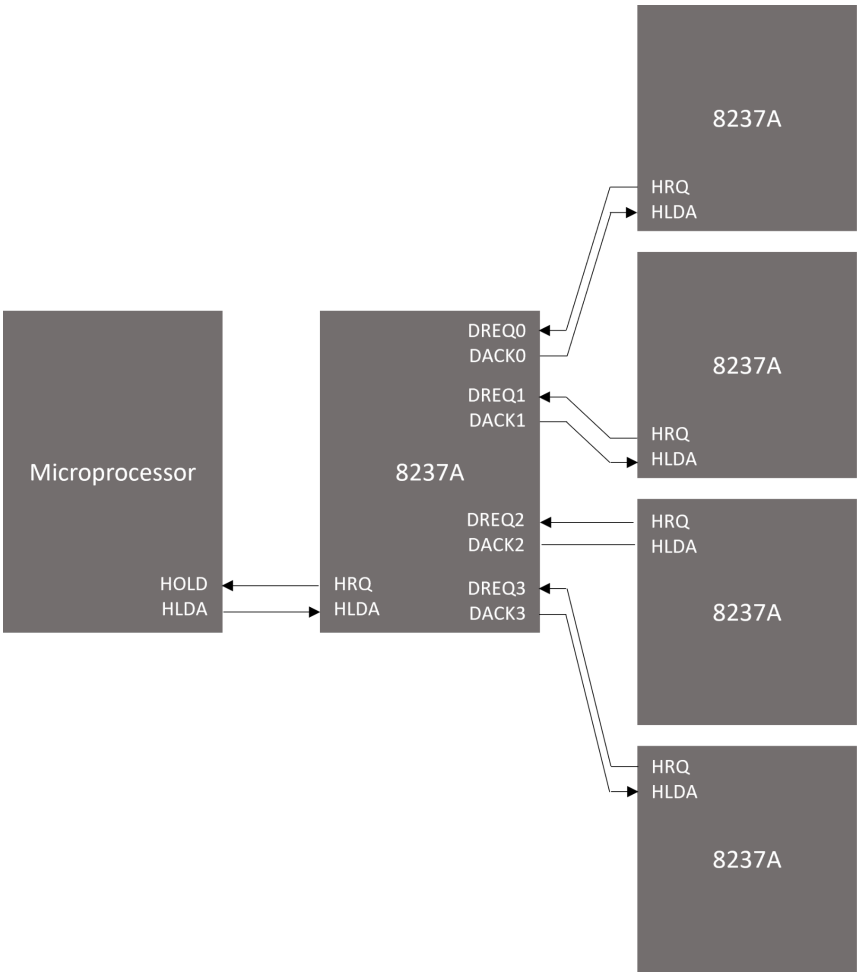
IN AX, DX / IN AL, DX / OUT DX, AL

2. (a) To adopt the utility of 15 channels, estimate the minimum number of secondary 8237 DMA controllers needed.

⇒ Each master controller can host 4 slave controllers as the 8237 DMA controller has 4 channels. Thus, for 15 channels in total, a system will require at least 4 slaves (secondary controller)

(b) Construct the DMA cascading diagram based on your answer in (a). Your diagram should show how the secondary controllers are connected with the primary controller using the appropriate pins.

⇒



3. Describe the values of the given pins of an 8237 DMA Controller during a DMA write cycle:

i. $\overline{IOR}$ ii. $\overline{IOW}$ iii. $\overline{MEMW}$ iv. $\overline{MEMR}$

⇒ During a DMA Write cycle, the DMA will read data from a peripheral and write the data to memory. Thus the pin conditions during a write cycle will be:

i. $\overline{IOR}$ ⇒ Low, since reading from peripheral

ii. $\overline{IOW}$ ⇒ High

iii. $\overline{MEMW}$ ⇒ Low, since writing data to memory

iv. $\overline{MEMR}$ ⇒ High

4. Describe the values of the given pins of an 8237 DMA Controller during a DMA read cycle:

i. $\overline{IOR}$ ii. $\overline{IOW}$ iii. $\overline{MEMW}$ iv. $\overline{MEMR}$

⇒ DMA will read data from memory and write the data to a peripheral device.

i. $\overline{IOR}$ ⇒ High
ii. $\overline{IOW}$ ⇒ Low, since writing to peripheral
iii. $\overline{MEMW}$ ⇒ High
iv. $\overline{MEMR}$ ⇒ Low, since reading from memory

5. For data transfer the following scenarios, among programmed I/O, Interrupt based I/O and DMA based I/O, which one would you prefer and why?

a. Matrix keypad connected to a door locking mechanism
⇒ We can use programmed I/O as matrix keypads are not required to be very high speed/ instantaneous.
b. Power outage detection in computers
⇒ Interrupt based I/O should be used so that the CPU can handle these special scenarios immediately.
c. Transferring data from Hard disk to RAM.
⇒ Requires high speed transmission in bulk without affecting processing speeds, so DMA should be used.

6. How many I/O ports does the 8086 processor support in the isolated I/O method?
⇒ 65536 (2^16)

7. A particular I/O device utilizes the MOV operation to take input from a temperature sensor. Is the device memory-mapped or isolated?
⇒ Using MOV operation implies memory mapped I/O
Using IN/OUT operation implies isolated I/O