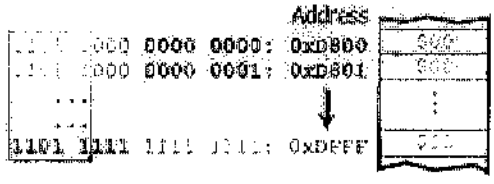
Answer of the Microprocessor final exam 2016-2017

Question number one [25 Marks]

1. A cache memory device is connected to a microprocessor has 16-bit address line. The microprocessor has assigned the addresses D800 to DFFF to this cache. What is the size of this memory device? What is the minimum number of bits required to represent the addresses only for this cache memory device? [4 Marks]

Solution:

The range D800 to DFFF is akin to all possible cases with 11 bits. Thus the cache memory size is 2 KB. The minimum number of bits required to represent the addresses for this cache is 11 bits.



2. Consider the binary numbers in the following addition and subtraction problems to be signed 6-bit values in the 2's complement representation. Perform each of the following operations, specifying whether overflow occurs. [6 Marks]

Solution:

011001	110111	011010	011010
+ 010000	+101011	- 100010	+ 011110
101001	100010		111000

a) Both numbers positive and a carry into the leftmost bit position: There is no carry out of the leftmost position, so the XOR is 1. The result has a sign bit =1, so there is over flow.
b) Both numbers negative and a carry into the leftmost bit position: There is a carry out of the leftmost position, so the XOR is 0. The result has a sign bit =1, so there is no over flow.
c) Both numbers positive and a carry into the leftmost bit position: There is no carry out of

3. Specify the four control signals commonly used by the 8085 microprocessor. What is the purpose of bus interface unit? [5 Marks]

Solution:

The control bus controls the memory and I/O, and requests reading or writing of data. Control is accomplished with $\overline{\text{IORC}}$ (I/O read control), $\overline{\text{IOWC}}$ (I/O write control), $\overline{\text{MRDC}}$ (memory read control), and $\overline{\text{MWTC}}$ (memory write control).

The bus interface unit controls the access to the system buses. The bus interface unit generates the memory address and control signals, and passes and fetches data or instructions to either a level 1 data cache or a level 1 instruction cache.

4. Identify the restrictions that were imposed on the values of the segment registers in real mode. Are the values placed in these registers by a program running in real mode useful under the same system running in protected mode? Why or why not? [2 Marks]

Solution:

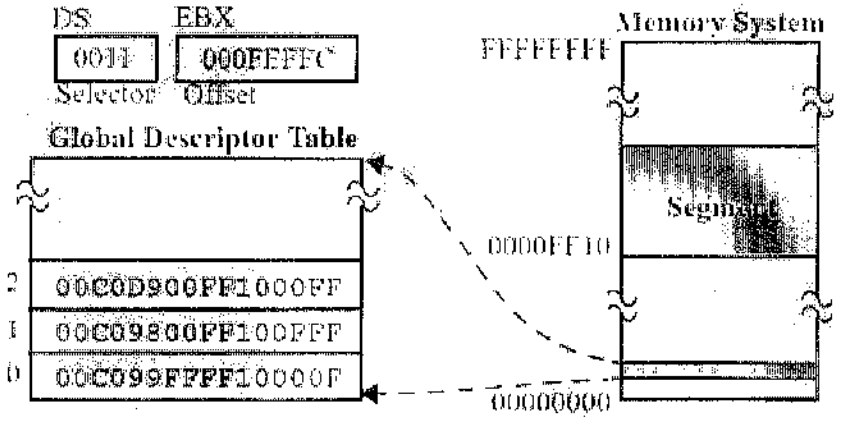
A segment register functions differently in the real mode when compared to the protected mode operation of the microprocessor. The segment address, located within one of the segment registers, defines the beginning address of any 64K-byte memory segment. The offset address selects any location within the 64K byte memory segment. When data and programs are addressed in protect mode, the offset address is still used to access information located within the memory segment. One difference is that the segment address, as discussed with real mode memory addressing, is no longer present in the protected mode. In place of the segment address, the segment register contains a selector that selects a descriptor from a descriptor table. The descriptor describes the memory segment's location, length, and access rights.

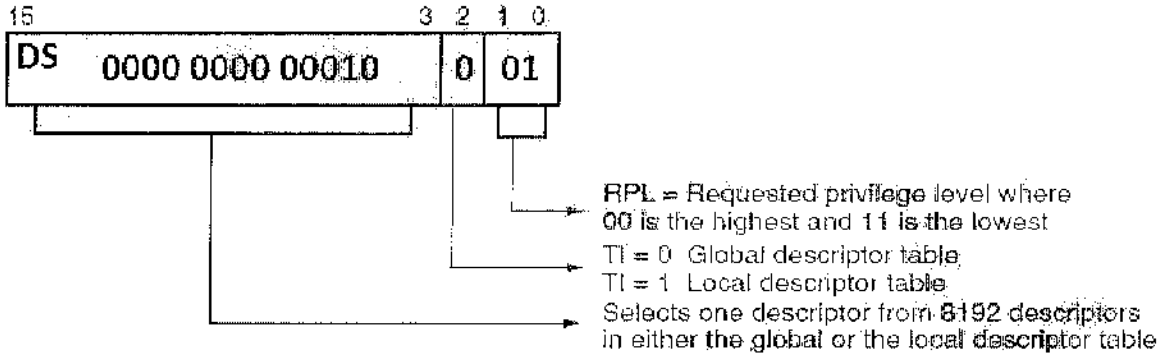
5. For a Core2 descriptor that contains a base of 00280000H, a limit of 00010H, and G = 1, what starting and ending locations are addressed by this descriptor? [3 Marks]

Solution:

The base address portion of the descriptor indicates the starting location of the memory segment (i.e., starting location is addressed by 00280000H). The limit contains the last offset address found in a segment. If G =1, the value of the limit is multiplied by 4K bytes (appended with FFF H). The limit is then 00010FFF H. Consequently, if a segment begins at memory location 00280000H and the value of limit is 00010FFF the ending location at location 00290FFFH.

6. Use the descriptor definition to decode the appropriate descriptor. [5 Marks]





Solution:

In this illustration, DS contains 0011H, which accesses the descriptor number 2 from the global descriptor table using a requested privilege level of 01. Descriptor number 2 contains a descriptor that defines the base address as 0000FF10H with a segment limit of 000FFH. This means that a value of 0011H loaded into DS causes the microprocessor to use memory locations 0000FF10H –0010FF0FH for the code segment with this example descriptor table.

Base = starting location of the memory segment = 0000FF10H

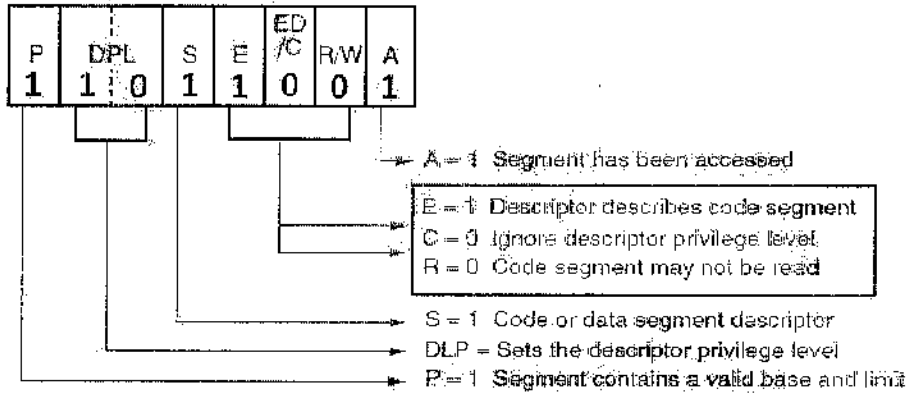
Limit = the last offset address found in a segment

G=1, the value of the limit is multiplied by 4K bytes (appended with FFF H).

The limit is then 000FFFFF H.

Ending location = Base + Limit = 0010FF0FH

Access rights = D9 H = 1101 1001



The offset address = 000FEFFC H that selects any location within 0000FF10 H – 0010FF0F H for the code memory segment.

1. Name the address mode used in the instructions below. Be specific, if for example the addressing mode is indexed, indicate which one in particular. Then, find the physical address in different mode (assume Real mode of operation). [9 Marks]
   a) MOV AX, [2500H]
   b) MOV AX,[BX]
   c) MOV [EAX+EBX], CL

Solution:

a) MOV AX, [2500H]

Direct Data Addressing:

Direct addressing with a MOV instruction transfers data between a memory locations located within the data segment and the AL, AX, or EAX. The address is formed by adding the displacement [2500H] to the default data segment address.

b) MOV AX,[BX]

Register Indirect Addressing:

Register indirect addressing allows data to be addressed at any memory location through an offset address held in any of the following registers: BP, BX, DI, and SI. The data segment is used by default with register indirect addressing or any other addressing mode that uses BX, DI, or SI to address memory. The address is formed by adding the contents in BX to the default data segment address.

c) MOV [EAX+EBX], CL

Base-plus-index addressing

Base-plus-index addressing is similar to indirect addressing because it indirectly addresses memory data. In the 80386 and above, this type of addressing allows the combination of any two 32-bit extended registers except ESP. In the given example, the MOV [EAX+EBX], CL instruction is using EBX (as the base) plus EAX (as the index). The address is formed by adding the contents in EBX plus the contents in EAX to the default data segment address.

2. What is the purpose of the direction flag? Which instructions set and clear the direction flag? Which string instruction use both DI and SI to address memory data? Then, explain the operation of the STOSW instruction. [8 Marks]

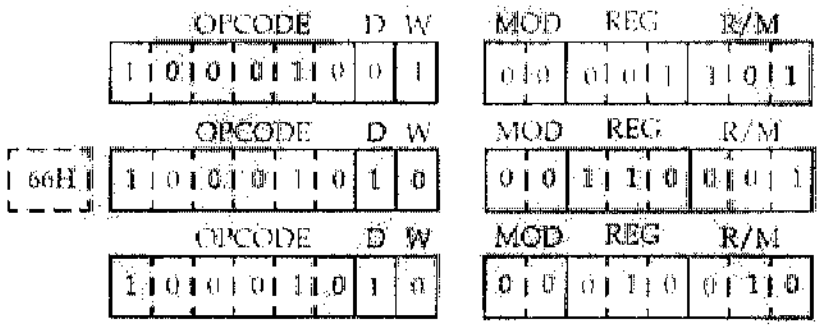
Solution:

The Direction Flag (D, located in the flag register) selects the auto-increment or the auto-decrement operation for the DI and SI registers during string operations. The CLD instruction clears the D-flag and the STD instruction sets it. Whenever a string instruction transfers a byte, the contents of DI and/or SI are incremented or decremented by 1. If a word is transferred, the contents of DI and/or SI are incremented or decremented by 2. Double-word transfers cause DI and/or SI to increment or decrement by 4.

The MOVS instruction use both DI and SI to address memory data segment. The MOVS instruction transfers a byte, word, or double-word from the data segment location addressed by SI to the extra segment location addressed by DI. As with the other string instructions, the pointers then are incremented or decremented, as dictated by the direction flag.

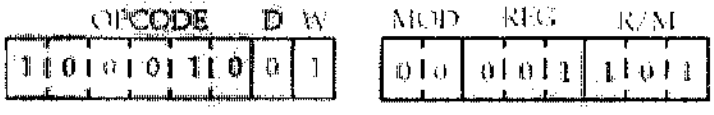
The STOS instruction stores AL, AX, or EAX at the extra segment memory location addressed by the DI register. STOS instruction may be appended with a B, W, or D for byte, word, or double-word transfers. The STOSW (stores a word) instruction stores AX in the extra segment memory location addressed by DI. After the byte (AL), word (AX), or double-word (EAX) is stored, the contents of DI increment or decrement.

3. Describe the purpose of the D- and W-bits found in some machine language instructions. After that, give the instruction encoded by the following if operating in 32-bit mode: [8 Marks]

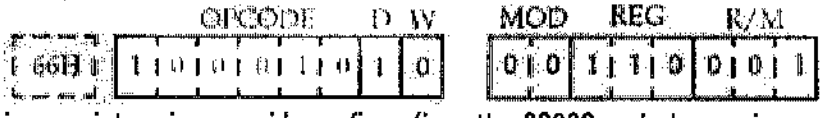


Solution:

If the direction bit (D-bit) =1, data flow to the REG field from the R/M field located in the second byte of an instruction. If the in the opcode D =0, data flow to the R/M field from the REG field. If the W=1, the data size is a word or double-word; if W =0, the data size is always a byte. The instruction mode and register-size prefix (66H) determine whether W represents a word or a double-word.



The opcode = 100010 (MOV instruction)
D = 0 (data flow to the R/M field from the REG field).
W = 1 (the data size is a double-word).
MOD = 00 (R/M field selects memory-addressing modes without displacement)
REG = ECX
R/M = [EBP]
MOV [EBP], ECX



66 H is a register size override prefixes (i.e., the 80386 and above microprocessors are operated in the 16-bit instruction mode).
The opcode = 100010 (MOV instruction)
D = 1 (data flow to the REG field from the R/M field).
W = 0 (the data size is a byte).

MOD = 00 (R/M field selects memory-addressing modes without displacement)
REG = EDX
R/M = [BX+DI]
MOV EDX, [BX+DI]

The opcode = 100010 (MOV instruction)
D = 1 (data flow to the REG field from the R/M field).
W = 0 (the data size is a byte).
MOD = 00 (R/M field selects memory-addressing modes without displacement)
REG = DL
R/M = [EDX]
MOV DL, [EDX]

4. Show an operation that LEA can perform that MOV cannot. [5 Marks]
   a) LEA BX, [DI]
   b) MOV BX, DI

Solution:

By comparing LEA with MOV, we observe that LEA BX, [DI] loads the offset address specified by [DI] (contents of DI) into the BX register; MOV BX, [DI] loads the data stored at the memory location addressed by [DI] into register BX. Suppose that the microprocessor executes an LEA BX, [DI] instruction and DI contains a 1000H. Because DI contains the offset address, the microprocessor transfers a copy of DI into BX. A MOV BX, DI instruction performs this task in less time and is often preferred to the LEA BX, [DI] instruction.

5. Describe how the LDS BX, NUMB instruction operates. [5 Marks]

Solution:

This instruction transfers the 32-bit number, addressed by symbolic data segment memory location NUMB in the data segment, into the BX and DS registers. The offset address appears first, followed by the segment address (i.e., LDS instruction loads BX with the word stored at data segment memory location NUMB and DS is loaded from the data segment memory location addressed by NUMB+2).