CS401 Short Notes

By

http://vustudents.ning.com

Op-Code:

An op-code (operation code) is the portion of a machine language instruction that specifies the operation to be performed. For every specific operation like add subtract or multiplication there is an op-code which processor understands. Like "add" we have 152 and subtract "153".

Program Counter/ Instruction Pointer:

A Program is an ordered set of instructions. Instruction executes one after the other. Program Counter / Instruction Pointer ensure that instructions are working in order also Program counter/Instruction Pointer holds the address of the next instruction to be executed.

Control Bus:

It's the part of computer Bus. CPU communicates with other devices within the computer through control bus. Through control bus CPU assign different jobs to other parts (registers, memory, input output) to perform some functionality on data.

For Example
Read data from memor
Write data on memory
Move data in register.

Control Lines:

As CPU have to perform more than one control operations so control bus consist of more than one lines called control lines.

The number and type of lines in a control bus varies but there are basic lines common to all microprocessors,

Read
Write

Offset

A distance from a given paragraph boundary in memory. The offset usually is given as a number of bytes.

Segment Registers:

In certain processors memory is segmented (or divided).Its actually not physically segmented rather logically to store code, data, stack etc in different portions of memory. So there should be some way to specify the starting address of each such segment. That's way registers are used called segment registers.

There are four types of these registers
CS: Code Segment
DS: Data Segment
SS: Stack Segment

ES: Extra Segment

CS contains the segment of the current instruction (IP is the offset).
DS is the segment used by default for most data operations or most memory references come from data segment.
SS contains the stack segment (SP is the offset).
ES as name shows extra segment register. String instructions(movs,stos) also use this register.

Logical Address and Physical Address:

The segment, offset pair is called a logical address, while the 20bit address is a physical address which is the real thing. Logical addressing is a mechanism to access the physical memory.

Offset:

A distance from a given paragraph boundary in memory. The offset usually is given as a number of bytes

Word is unit of data, like (bit -1 , nibble -4 , byte-8, word-16, Dword-32, Qword-64)
A word is of 2 bytes = 16bits
Dword = 4 bytes
Qword= 8 bytes

Registers are like **Scratch Pad Ram**.What are the meaning of scratch pad ram. Scratchpad means a high-speed internal memory used for temporary storage of preliminary information.

An 8-bit register, like AH, AL, BH, BL, can only store one data element at a time. And a 16-bit register can store multiple data element e.g. one in AH and other in AL.

Segment memory model or segment register is all about how processor sees the different portion of the memory. Our Program is consisting of two parts, the code and the data, and actually there is a third part called the program stack as well, but higher level languages make this invisible to us. These three logical parts of a program should appear as three distinct units in memory, but making this division is not possible in the linear memory model. The segmented memory model does allow this distinction. We divide the memory into the functional parts or windows, one is for code and other is for data. Processor sees code form the code window and data from the data window. Now if process sees the code section it cant see the data section through the same window and vice versa. At a time four window exist one is CS(Code window) hold the base point of the code 2nd is DS(Data window) hold the base point of the data 3rd is SS(Stack segment) holds the base point of the stack segment and last one is ES(Extra segment) its use only for data movement. Size of each window is 64K but it can move in any region in

1MB size. Segment memory model is all about how processor sees the different portion of the memory. I hope you can now understand the segmented memory model.

This problem is discussed in P.14 of the handouts. Just read the handout with full concentration. Here the Segment and Offset address are given which are necessary to make physical address. Both offset and segment address are 16 bit to make 20 bit we use mechanism mention below.
Now as Figure Show

Segment register = 1DDD it is16 bit to make 20 bit we add 0 with the end of the segment register now it become = 1DDD0
And offset 0100 which is also 16 bit now we add 0 before the offset address to make 20 bit. Offset = 0100 after append 0 before it become 00100

Now add both

```
 1DDD0
 00100  +
 ------
 1DED0   that is 20 bit physical address.
```

I hope now you understand how to calculate physical address.

Op-code

The op-code or operation code is the binary pattern that represents an instruction.

Segmentation:

The segmented memory model allows multiple functional windows into the main memory, a code window, a data window etc. The processor sees code from the code

window and data from the data window. The size of one window is restricted to 64K. However the maximum memory iAPX88 can access is 1MB which can be accessed with 20 bits.

Mnemonic

A word, abbreviation, or acronym that replaces something too complex to remember or type easily. For example, ADC is the mnemonic for the 8086's add-with-carry instruction. The assembler converts it into machine (binary) code, so it is not necessary to remember or calculate the binary form.

Big-endian and little-endian are terms that describe the order in which a sequence of bytes are stored in computer memory. Big-endian is an order in which the "big end" (most significant value in the sequence) is stored first (at the lowest storage address). Little-endian is an order in which the "little end" (least significant value in the sequence) is stored first. For example, in a big-endian computer, the two bytes required for the hexadecimal number 4F52 would be stored as 4F52 in storage (if 4F is stored at storage address 1000, for example, 52 will be at address 1001). In a little-endian system, it would be stored as 524F (52 at address 1000, 4F at 1001).

IBM's 370 computers, most RISC-based computers, and Motorola microprocessors use the big-endian approach.On the other hand, Intel processors (CPUs) and DEC Alphas and at least some programs that run on them are little-endian.

Shift Operation: In this operation, the digits are moved, or shifted, to the left or right. Registers in a computer processor have a fixed number of available bits for storing data, so during shift some bits will be "shifted out" of the register at one end, while the same number of bits are "shifted in" from the other end.shifted out bits are either discarded or preserved in Flags.

Rotation: Rotate instructions are similar to shift instructions, except that rotate instructions are circular, with the bits shifted out at one end returning(re-entring) on the other end. Rotates can be to the left or right.

What is CALL and RET function ?

In every processor, instructions are available to divert temporarily and to divert permanently. The instructions for permanent diversion in 8088 are the jump instructions, while the instruction for temporary diversion is the CALL instruction. The word call must be familiar to the readers from subroutine call in higher level languages. The CALL instruction allows temporary diversion and therefore reusability of code.

For example in the bubble sort code one place and reuse it again and again. This was not possible with permanent diversion. Actually the 8088 permanent diversion mechanism can be tricked to achieve temporary diversion. CALL takes a label as argument and execution starts from the label, until the RET instruction is encountered and it takes execution back to the instruction following the CALL. Both the instructions are mostly used as a pair. But they are independent CALL can be use independent of the RET and RET can also be use independent of the CALL. RET give control back to the next instruction after the CALL.

Here in this subject we have studied different "Intel processor architectures”. From these one of earlier architecture is Intel 8080 and other is Intel 8085.These architectures use memory modeling scheme called linear memory model or flat memory model. So A flat memory model uses a linear (or sequential) addressing scheme, allowing direct addressing of all available memory locations.

Now when a processor or DMA-enabled device needs to read or write to a memory location, it specifies that memory location on the address bus (the value to be read or written is sent on the data bus). The width of the address bus determines the amount of memory a system can address. As in Intel 8080 and 8086 processor internal registers are of 16 bits that’s why the maximum **linear** address space was limited to 64 KB.

When designing iAPX88 the Intel designers wanted to remain compatible with 8080 and 8085 however 64K was too small to continue with, for their new processor. To get the best of both worlds they introduced the segmented memory model in 8088. There is also a logical argument in favor of a segmented memory model in addition to the issue of compatibility discussed above. We have two logical parts of our program, the code and the data and actually there is a third part called the program stack as well, but higher level languages make this invisible to us. These three logical parts of a program should appear as three distinct units in memory, but making this division is not possible in the linear memory model. The segmented memory model does allow this distinction.
Intel AiAPX88 is also architecture of 16 bits processor but in order to access segmented memory it will generate the address of 20 bits as follows

```
Address = 16*segment + offset
```

And 20-bit external address bus gave an 1 MB (segmented) physical address space (2^{20} = 1,048,576).

The **SCAS** stand for Scan String. Basically it’s a block processing instruction in Intel 8088. SCAS compares a source byte or word in register AL or AX with the destination string element addressed by ES:DI and updates the flags. DI is updated to point to the next location. SCAS is often used to locate equality or in-equality in a string through the use of an appropriate prefix. SCAS is a bit different from the other instructions. This is more like the CMP instruction in that it does subtraction of its operands. The prefixes REPE (repeat while equal) and REPNE (repeat while not equal) are used with this instruction. The instruction is used to locate a byte in AL in the block of memory. When the first equality or inequality is encountered; both have uses. For example this instruction can be used to search for a 0 in a null terminated string to calculate the length of the string. In this form REPNE will be used to repeat while the null is not there.

LES and LDS Instructions

As Mention in the lecture that the string instructions need their source and destination in the form of a segment offset pair, there are two special instructions that load a segment register and a general purpose register from two consecutive memory locations. LES loads ES while LDS loads DS. These instructions have two parameters, one is the general purpose register to be loaded and the other is the memory location from which to load these registers. According to Intel rules of significance the word at higher address is loaded in the segment register while the word at lower address is loaded in the offset register. As parameters segment should be pushed first so that it ends up at a higher address and the offset should be pushed afterwards. When loading the lower address will be given.

far jump is one that uses the full segment base: offset value as an absolute address. In far jump in addition to the simple jump operations, there are the call (call a subroutine) and ret (return from subroutine) instructions. Before transferring control to the subroutine, call pushes the segment offset address of the instruction following the call onto the stack; ret pops this value off the stack, and jumps to it, effectively returning the flow of control to that part of the program. In the case of a far call, the segment base is pushed following the offset; far ret pops the offset and then the segment base to return from that subroutine.

So from above explanation you can understand that the example of far jump is calling a subroutine and then retuning from that.

There are three types of unconditional jumps
Short jumps
Near jumps
Far jumps
Their range calculation is as follows
The **short jump** is 2-byte instructions allow jumps or branches to memory locations with in +127 and -128 bytes from the address following the jump. This is the limitation of a byte in singed representation.
Near jump is a 3-byte instructions allow a branch or jump within ±32K bytes (or anywhere in the current segment) from the instructions in the current code segment.
Finally the 5-byte **far jump** allows a jump to any memory location within the real memory system
The short and near jumps are often called intra segment jumps and far jumps are often called inter segment jumps

Signed numbers are represented in form of 2's compliment in memory and registers and these are calculated as follows
fisrt take 1's compliment of a number
then add "1" in that complimented number and
the result will show the number in 2's compliment or signed number.