Lab 13

Title:

Addressing Modes in Assembly Language

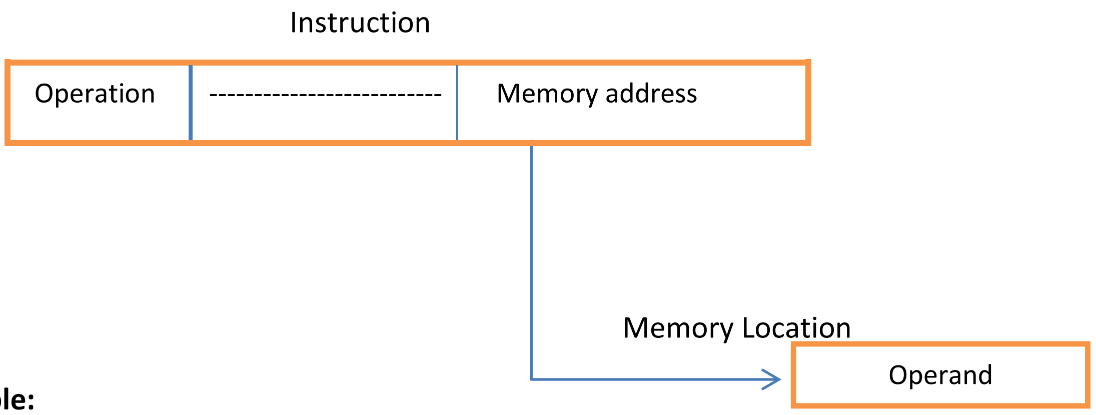
Introduction:

Here we are going to discuss two addressing modes

- Immediate Addressing
- Direct Addressing
- Indirect Addressing

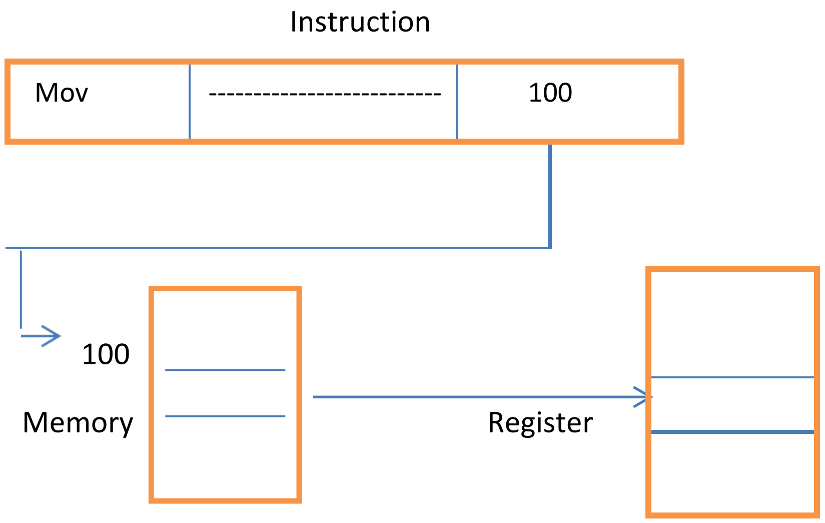
Direct Addressing Mode:

The operand is in memory and the memory address of the operand is held in instruction.



Example:

Mov ax, [100]



While it might be possible to write programs that used numeric addresses as operands, it is much easier to use symbolic names such as Var1.The assembler automatically convert a name such as var1 to its numeric offset and then differences the offset.

```
.data
 Var1   DWORD 30000h
.code
Moveax, var1
Call DumpRegs
.exit
```

Direct – Offset operands:

You can add a displacement to the name of a variable, creating a direct –offset operand. This lets you access memory location that may not have explicit label.

```
Example:
.data
array BYTE  10h,30h,45h,13h
.code
moval,array ;al=10h
mov al, [array+1]                               ; al=30h
mov al, [array+2]                                ;al=45h
mov al, [array+3]                                ;al=13h
.exit
```

An expression array+1 produce what is called an effective address by adding a constant to the variable offset. When we surround the effective address with brackets, it is to show that expression is dereferenced to obtain the contents of target memory location.

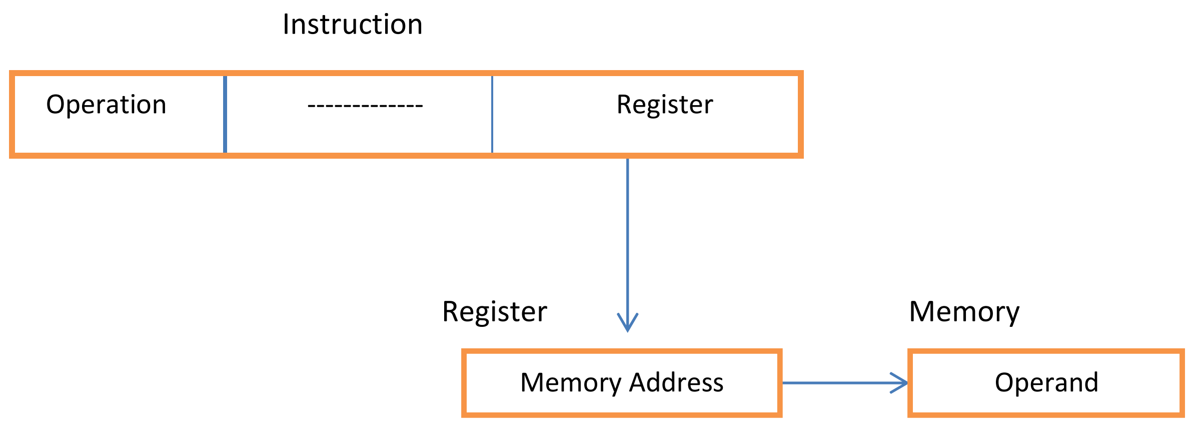
Range checking: MASM has no built-in range checking for effective addresses. According to above example if we write following statement

mov al , [array+20]

The assembler just retrieves a byte of memory from outside of array and will create hard-to-find error.so in this case programmers try to be careful when dealing with arrays.

Indirect Addressing Mode:

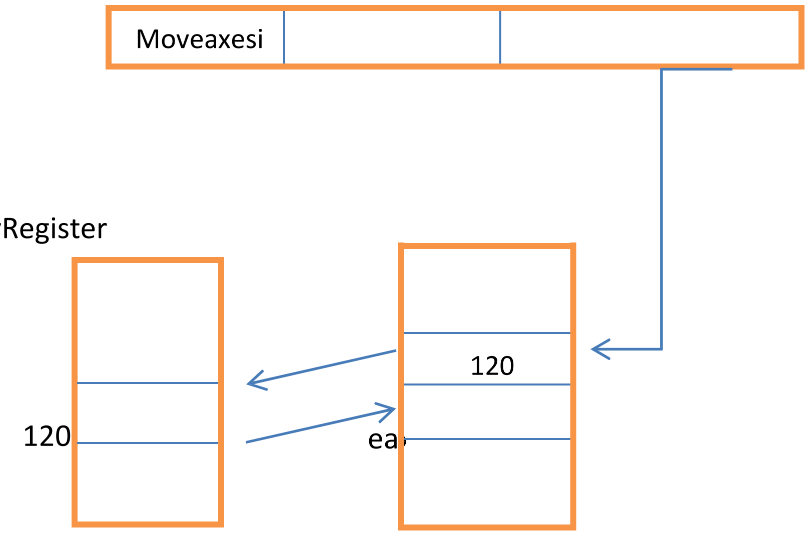
Direct addressing is completely impractical for array processing. We would never provide a different label name for every element of an array. The only practical way to handle an array is to use a register as a pointer and find ways to manipulate the register's value. This is called indirect addressing.



Example:

Moveax,[esi]

MemoryRegister



<u>Indirect Operands</u>

Indirect operands can be any 32-bit general purpose register (EAX, EBX, ECX, EDX, ESI, EBP and ESP) .The register is assumed to contain the Offset of some data.

Example:

```
.data
Array BYTE  10h,30h,20h
.code
Movesi,OFFSET array
Mov  al,[esi]
Incesi
Mov al,[esi]
```

```
Incesi
Mov al,[esi]
Incesi
.exit
```

Indirect operand is useful when dealing with arrays because an indirect operand value can easily be modified. An indirect operand can point to different array elements. In above example array contains three bytes and increment esi and make it point to each byte.

Lab Tasks

1. Write a program that will perform following tasks using procedure from the link library.
    - **a.** Input 32-bit integer from user and display it in decimal, hexadecimal and binary.
    - **b.** Display CPU Registers
2. Generate 10 random numbers between 0 and 20.Every time program should display different random numbers.
3. Ask the user to input their name and redisplay it.
4. Write a program that swaps two 16 bit numbers stored in memory.
5. Write instructions that use direct-offset addressing to move the four values in Uarray to the EAX, EBX, ECX, and EDX registers. When you follow this with a call DumpRegs statement, the following register values should display:

    Uarray WORD 1000h, 2000h, 3000h,4000h