COMPUTER INSTRUCTIONS

By: Dr. Yogendar Kumar Verma

INSTRUCTION FORMAT

- An instruction format or instruction code group of bits used to perform a particular operation on the data stored in computer.
- Processor fetches an instruction from memory and decodes the bits to execute the instruction.
- Different computers may have their own instruction set.

INSTRUCTION FORMAT

- Instruction code is divided into two parts namely operation code and address of data.
- Operation code consisting group of bits to define an operation such as add, subtract, multiply etc.

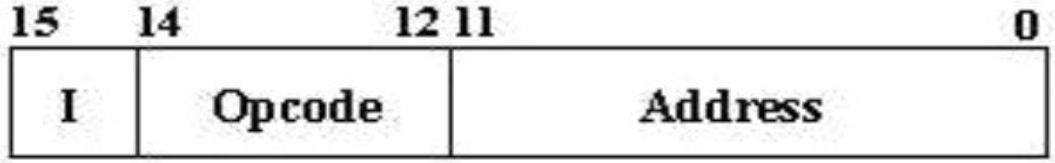
INSTRUCTION FORMAT

In an instruction format:

- First 12 bits (0-11) specify an address.
- Next 3 bits specify operation cod (opcode).
- Left most bit specify the addressing model

I = 0 for direct address

I =



Instruction format

TYPES OF INSTRUCTIONS

The basic computer has three 16-bit instruction code formats:

1. Memory Reference Instructions
2. Register Reference Instructions
3. Input/Output Instructions

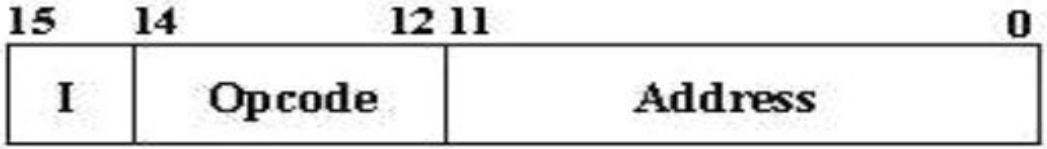
Memory Reference Instructions

In Memory reference instruction:

- First 12 bits(0-11) specify an address.
- Next 3 bits specify operation code (opcode).
- Left most bit specify the addressing mode I

I = 0 for direct address

I = 1 for indirect address



Instruction format

(Opcode = 000 through 111)

Memory Reference Instructions

In Memory reference instruction:

- first 12 bits (0-11) specify an address.
- The address field is denoted by three x's (in hexadecimal notation) and is equivalent to 12-bit address.
- The last mode bit of the instruction represents by symbol I.
- When I = 0, the last four bits of an instruction have a hexadecimal digit equivalent from 0 to 6 since the last bit is zero (0).
- When I = 1 the last four bits of an instruction have a hexadecimal digit equivalent from 8 to E since the last bit is one (1).

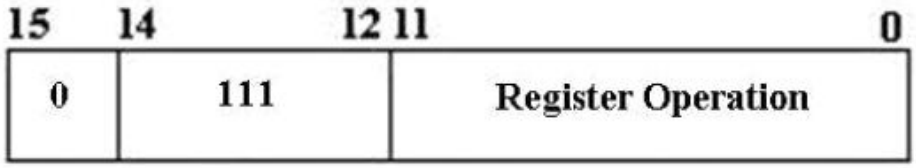
Memory Reference Instructions

	Hexadecimal code		
Symbol	I = 0	I = 1	Description
AND	0xxx	8xxx	AND memory word to AC
ADD	1xxx	9xxx	ADD memory word to AC
LDA	2xxx	Axxx	LOAD Memory word to AC
STA	3xxx	Bxxx	Store content of AC in memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and Skip if zero

Register Reference Instructions

In Register Reference Instruction:

- First 12 bits (0-11) specify the register operation.
- The next three bits equals to 111 specify opcode.
- The last mode bit of the instruction is 0.
- Therefore, left most 4 bits are always 0111 which is equal to hexadecimal 7.



Instruction format

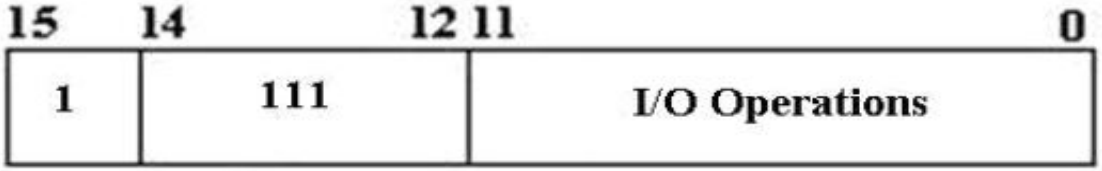
Register Reference Instructions

Symbol	Hexadecimal code	Description
CLA	7800	Clear AC
CLE	7400	Clear E
CMA	7200	Complement AC
CME	7100	Complement E
CIR	7080	Circulate right AC and E
CIL	7040	Circulate left AC and E
INC	7020	Increment AC
SPA	7010	Skip next instruction if AC positive
SNA	7008	Skip next instruction is AC is negative
SZA	7004	Skip next instruction is AC is 0
SZE	7002	Skip next instruction is E is 0
HLT	7001	Halt computer

I/O Reference Instructions

In I/O Reference Instruction:

- First 12 bits (0-11) specify the I/O operation.
- The next three bits equals to 111 specify opcode.
- The last mode bit of the instruction is 1.
- Therefore, left most 4 bits are always 1111 which is equal to hexadecimal **F**.



Instruction format

I/O Reference Instructions

Symbol	Hexadecimal code	Description
INP	F800	Input character to AC
OUT	F400	Output character from AC
SKI	F200	Skip on input flag
SKO	F100	Skip on Output flag
ION	F080	Interrupt on
IOF	F040	Interrupt off

Addressing Modes

Addressing mode provides different ways for accessing an address to given data to a processor. Operated data is stored in the memory location, each instruction required certain data on which it has to operate. There are various techniques to specify address of data. These techniques are called Addressing Modes.

- **Direct addressing mode** – In the direct addressing mode, address of the operand is given in the instruction and data is available in the memory location which is provided in instruction. We will move this data in desired location.
- **Indirect addressing mode** – In the indirect addressing mode, the instruction specifies a register which contain the address of the operand. Both internal RAM and external RAM can be accessed via indirect addressing mode.
- **Immediate addressing mode** – In the immediate addressing mode, direct data is given in the operand which move the data in accumulator. It is very fast.
- **Relative addressing mode** – In the relative address mode, the effective address is determined by the index mode by using the program counter in stead of general purpose processor register. This mode is called relative address mode.
- **Index addressing mode** – In the index address mode, the effective address of the operand is generated by adding a content value to the contents of the register. This mode is called index address mode.

2.1 Direct Addressing

When the instruction explicitly states the location of an operand or a destination (either in memory or in a processor register), the addressing mode is known as *direct addressing*. The effective address itself is included in the subsequent words of the instruction *(post-words)*.

Two subclassifications within the direct addressing mode are often recognized. When the location is in memory the mode may be referred to as *absolute addressing*. When the location is a processor register it may be referred to as *register direct addressing*. Figure 5.3 illustrates the two modes. In part *a* the instruction specifies the address of the operand; in part *b* the instruction specifies the register containing the operand.

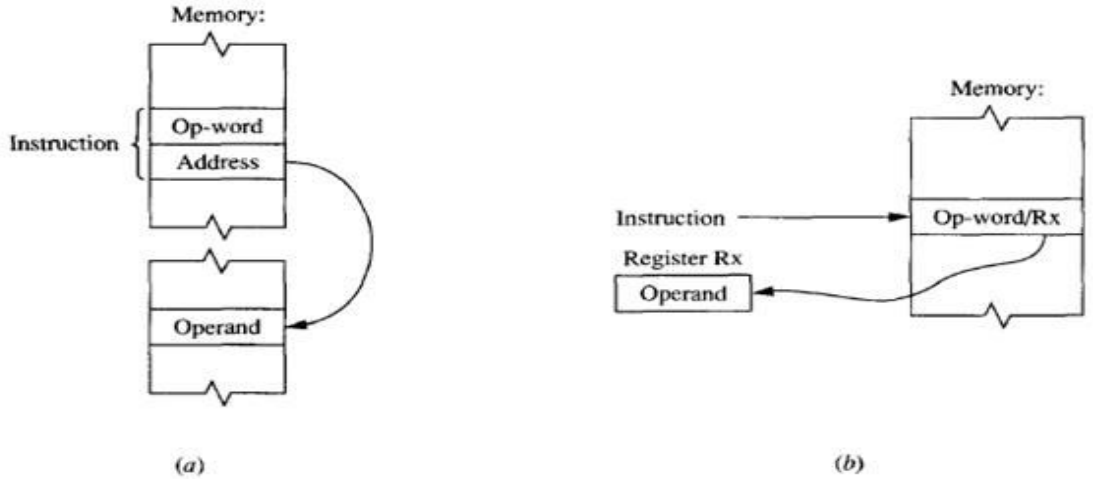
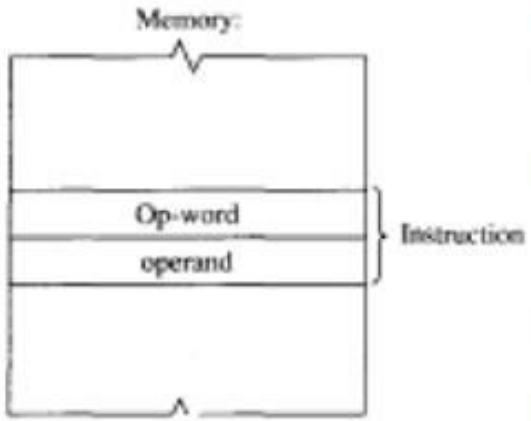


FIGURE 5.3 Direct addressing: (a) absolute addressing, *(b)* register direct addressing.
(LDA 100BH, ADDA 100CH, STA 100DH

Direct addressing is used when the memory address or the selected register is to be fixed in the program.

2.2 Immediate Addressing

In many cases an instruction requires a constant quantity, a bit pattern which will never change no matter when or how often the instruction is executed. This mode of including a bit pattern as a part of an instruction is called the *immediate addressing* mode. The op-word for the instruction includes a group of bits which identifies this mode of addressing, and the post-words include the bit pattern itself. Since the instruction is located in program memory the constant itself is also in program memory.



The immediate addressing mode the instruction does not state explicitly the location of the operand; rather, it explicitly states the operand itself. The example in Figure 5.4 illustrates this mode. Note that the operand becomes an integral part of the instruction.

FIGURE 5.4 Immediate addressing.

Immediate addressing is used when a particular constant value is to be fixed within the program itself. The value is found in memory "immediately" after the instruction code word and may never change at any time.

2.3 Indirect Addressing

In the *indirect addressing* mode the instruction tells the processor neither the address of the operand nor the operand itself. Instead, it tells the processor where to go to find the address of the operand. The instruction may explicitly state either the address of a location in memory or the name of a processor register, but the binary number which is found there is not the operand. Instead, it is the effective address, the address of a location in memory to which the processor must go to find the operand. The result is that the processor must take one extra step in order to locate the operand.

The op-word for the instruction includes a group of bits which identifies this mode of addressing, and the (indirect) address is specified in one or more additional post-words. If the instruction names a processor register as the source of the effective address, then the register identification number may fit into the op-word itself.

Indirect addressing is used when a program must operate upon different data values under different circumstances.

Figure 5.6 compares and contrasts the first three addressing modes. Note that in the immediate mode the instruction includes the operand, in the direct mode it includes the address of the operand, and in the indirect mode it includes the address of the address of the operand.

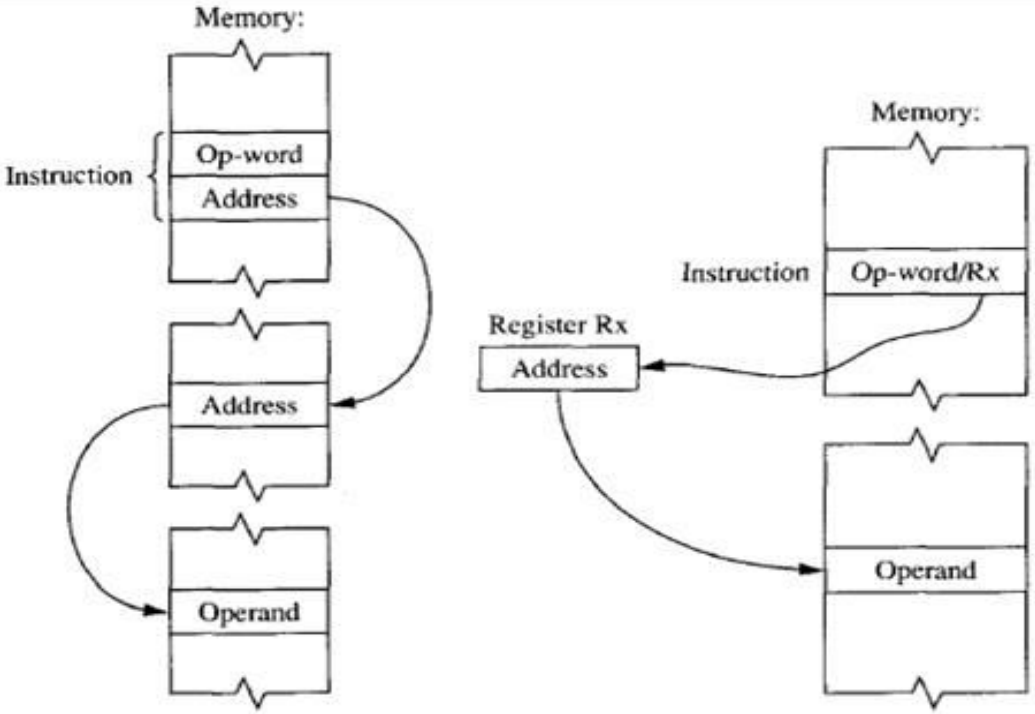
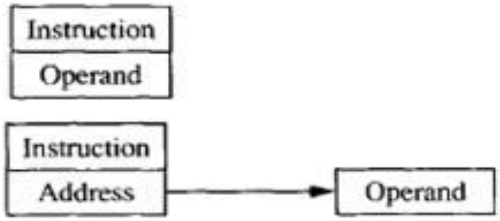


FIGURE 5.5 Indirect addressing: (a) memory indirect, *(b)* register indirect.



2.4 Multi-Component Addressing Modes

Each of the following related addressing modes requires that the processor assemble two or more components together during the execution of the program in order to create the effective address. In each case the effective address itself is that of a memory location. However, at least one of the components is found in a processor register.

Instructions which use *indexed addressing* specify two registers, often by coding within the op-word itself and known as "indexed addressing". During program execution the processor temporarily adds the contents of these registers to generate the effective address. One of the registers is an address register and it is said to hold the *base address*. The other is commonly a data register - the *displacement* or *index* register.

Based addressing is a similar mode wherein the instruction specifies an address register and a fixed constant (an *offset* or *displacement).* The register designation often fits within the op-word and the offset usually requires post-words. In this mode the content of the register is the base and the constant is the displacement. During execution the processor adds the constant and the value in the register to generate the effective address. This addressing mode is also known as "relative based indexed" mode

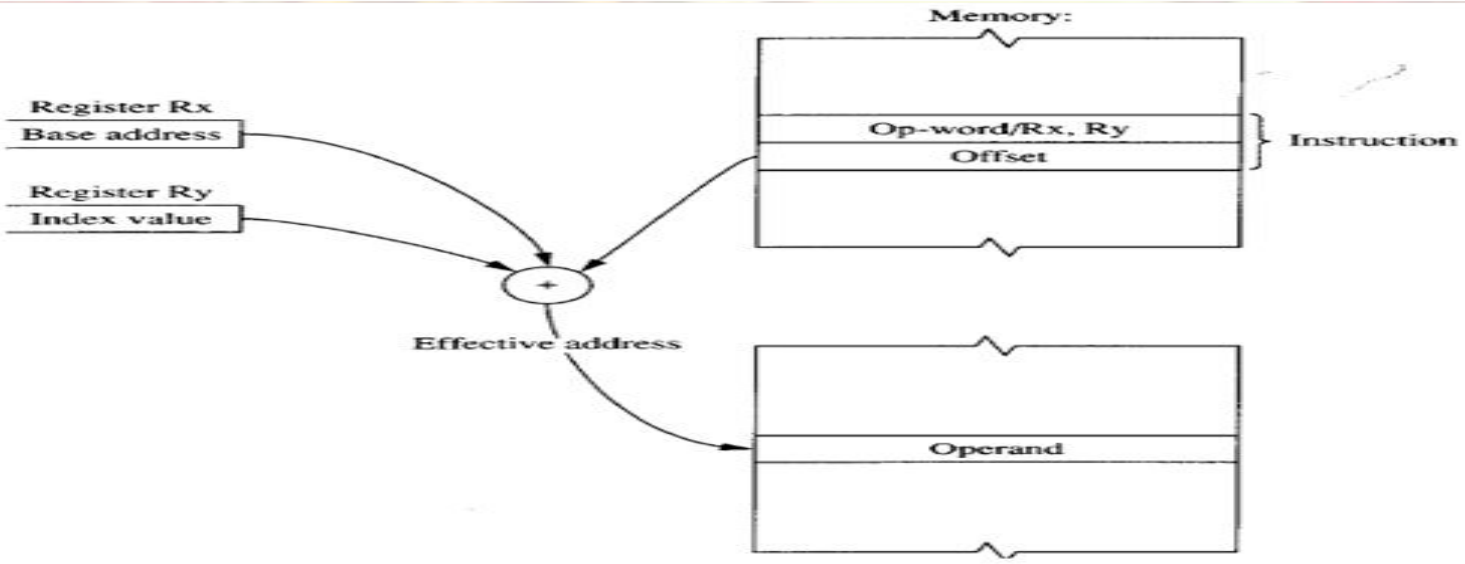
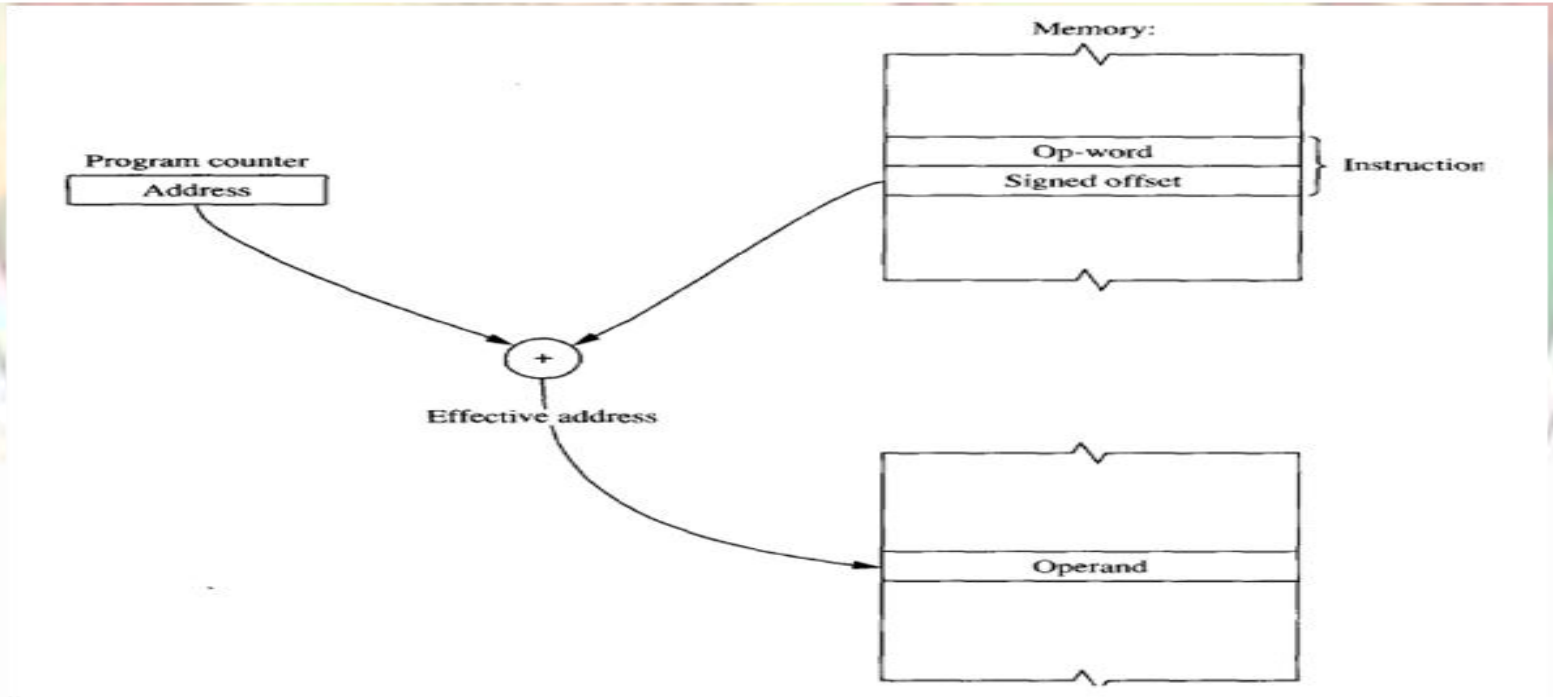


FIGURE 5.7 "Indexed/based/offset" addressing.

Relative Addressing Mode

The relative addressing mode permits the writing of "position-independent code," programs which will be properly executed by the processor regardless of where they are located in memory. The entire program (together with any necessary data) may be picked up from one region of memory and moved to another with no adverse effect. In order to be location-independent a program may not refer to any specific location by address. All references to memory must be through the use of relative addressing.



Thanks you…….