3.1 ARITHMETIC INSTRUCTIONS

The basic assembly language expression for arithmetic instruction is

Opcode Rd,Rn,Rm

Where the operation specified by the OP code is performed using the operands in general purpose registers Rn and Rm. The result is placed in register Rd.

Example:

ADD R0,R2,R4

performs the operation

R0←[R2]+[R4].

SUB R0,R6,R5

Performs the operation

R0←[R6]+[R5]

Instead of being contained in Rm , the second operand can be given in the immediate mode. Thus

ADD R0,R3,#17

performs the operation

R0←[R3]+17

The immediate value is contained in the 8 bit field in bits b_{7-0} of the instruction.

The second operand can be shifted or rotated before being used in the operation. as shown below

ADD R0,R1,R5 LSL #4

In the above example the second operand which is specified in register is shifted left 4 bit positions and is then added to the contents of register R1 and sum is placed in register R0.

MULTIPLY

There are two versions of multiply. The first version multiplies the contents of two registers and place the lower order 32-bits of the product in a third register. The higher order bits of the product, if there are any discarded.

For Example , The instruction

MUL R0,R1,R2

Performs the operation

R0←[R1]*[R2]

The second version specifies a fourth register whose contents are added to the product before storing the result in the destination register, hence the instruction

MLA R0,R1,R2,R3

R0←[R1]*[R2]+[R3];

This is called multiply accumulator operation. It is often used in numerical algorithm for digital signal processing.

3.2 LOGIC INSTRUCTIONS

The following are the logic instruction. These have the same format as arithmetic instructions.

1. AND
2. OR
3. XOR
4. BIC
5. MVN

AND Operation:

Format: AND Rd, Rn, Rm

Performs the operation

Rd←[Rn]^[Rm]

Which is a bitwise logical AND between the operands in register Rn and Register Rm.

Example:

R0=0110

R1=0011

Then the instruction

AND R0,R0,R1

Results in value

```
 0 1 1 0
 0 0 1 1
---------
 0 0 1 0
```

OR Operation:

Format: OR Rd, Rn, Rm

The bitwise logical OR performs OR operation over the contents of Rm and Rn and places the result in Rd.

Example:

R0=0110

R1=0011

OR R0, R0, R1

Results in value

```
 0 1 1 0
 0 0 1 1
---------
 0 1 1 1
```

XOR Operation:

In XOR operation when the two inputs of register are 0,1 or 1,0 then only it produce 1 otherwise it will produce 0.

Format: XOR Rd, Rn, Rm

Example:

R0=0110

R1=0011

Results in value

```
0 1 1 0
0 0 1 1
-------
0 1 0 1
```

MVN Operation:
Move negative operation complements the bits of the source operand and places the result in destination operand.

MVN Rd, Rs

Example:

MVN R0,R1

Where

R1=0010

It will result 1001 as output.

Example2:

MVN R0,R3

Here the contents of R3 are complemented and then moved to R0.

BIC(Bit Clear):
This instruction is used to clear the bits in a register.(NAND). It is equivalent to not AND operation.

Format:

BIC Rd, Rm, Rn

Example:

BIC R0, R1, R2

Let R1=0110

R2=0011 then

BIC R0, R1, R2 results in

```
0 0 1 1
0 1 1 0
-------
0 1 0 1
```

3.3 BRANCH INSTRUCTIONS

Conditional branch instruction contains a signed 2's complement, 24 bit offset that is added to update the contents of the program counter to generate the branch target address.

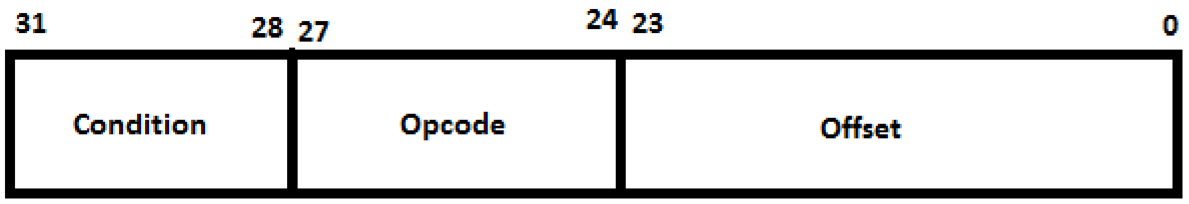


Fig:Instruction Format

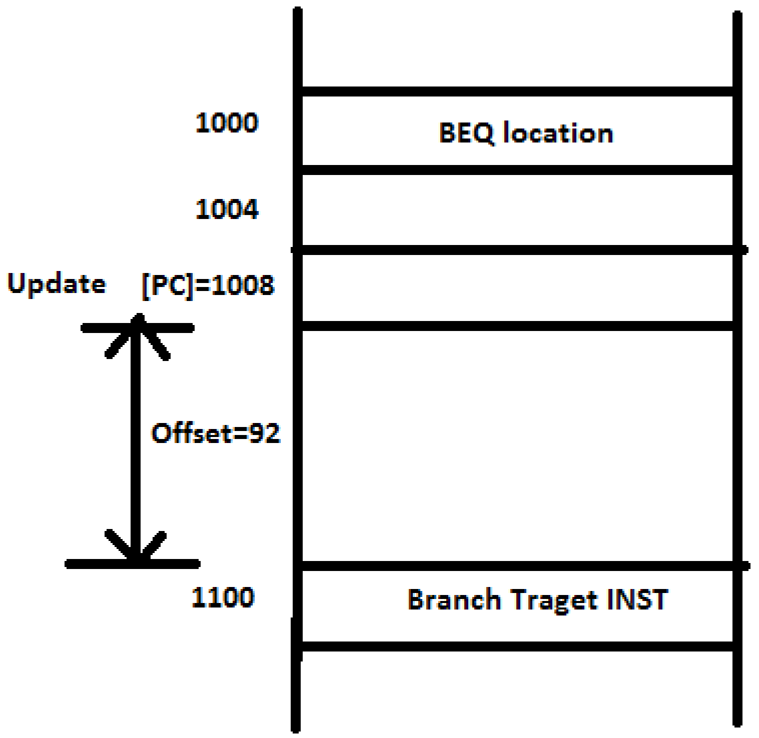


Fig: determination of branch target address

Fig: ARM Branch Instructions

The BEQ(Branch if equal to zero) causes a branch if Z(Zero) flag is set to 1. The condition to be tested to determine whether or not branching should takes place is specified in the higher order 4 bits(b31-28)of the instruction word.

At the same time that the branch target address is computed ,the content of PC have been updated to contain the address of the instruction that is two words beyond the branch instruction it self. If the branch instruction is at address location 1000, and the branch target address is 1100, then the offset has to be 92. Because the the contents of updated PC will be 1000+8=1008. When the address 1100 is computed

SETTING CONDITION CODES

Some instructions such as compare is given

Cmp Rn,Rm

Which performs the operation

[Rn] - [Rm]

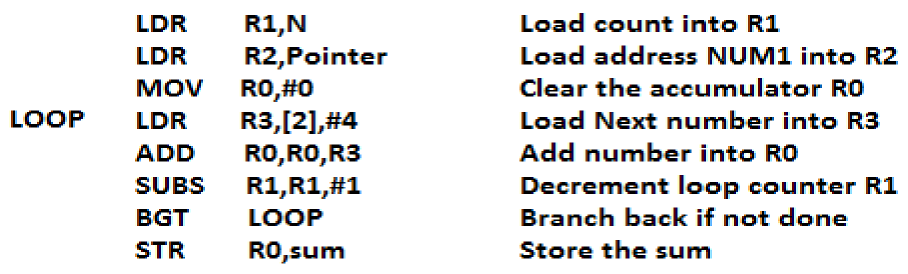
Have the sole purpose of setting the condition code flags based on the result of the subtraction operation. On the other hand, the arithmetic and logic instructions effect the condition code flags only if explicitly specified to do by a bit in the op code field. This is indicated by suspending the suffix S to the assembly language op code mnemonic.
For Example that, the instruction

ADDS R0,R1,R2

Sets the condition code flags but

ADD R0,R1,R2 does not.

A loop program for adding numbers



```
        LDR    R1,N              Load count into R1
        LDR    R2,Pointer        Load address NUM1 into R2
        MOV    R0,#0             Clear the accumulator R0
LOOP    LDR    R3,[2],#4         Load Next number into R3
        ADD    R0,R0,R3          Add number into R0
        SUBS   R1,R1,#1          Decrement loop counter R1
        BGT    LOOP              Branch back if not done
        STR    R0,sum            Store the sum
```

3.4 ADDRESSING MODES

The different ways in which the location of an operand is specified in an instruction is called as Addressing mode.

Generic Addressing Modes:

1. Immediate mode
2. Implied mode
3. Register mode
4. Direct mode or absolute mode
5. Indirect mode
6. Index mode
7. Base Register addressing mode
8. Relative mode
9. Auto-increment mode or decrement mode

Implementation of Variables and Constants:

Variables:

The value can be changed as needed using the appropriate instructions.

There are 2 accessing modes to access the variables. They are

- Register Mode
- Absolute Mode

Register Mode:
The operand is the contents of the processor register. The name(address) of the register is given in the instruction.

Absolute Mode(Direct Mode):
The operand is in new location. The address of this location is given explicitly in the instruction

Example: MOVE LOC,R2
The above instruction uses the register and absolute mode. The processor register is the temporary storage where the data in the register are accessed using register mode. The absolute mode can represent global variables in the program.

Mode	**Assembler Syntax**	**Addressing Function**
Register mode	Ri	**EA=Ri**
Absolute mode	LOC	**EA=LOC**

Where EA-Effective Address

Constants:
Address and data constants can be represented in assembly language using Immediate Mode.

Immediate Mode.
The operand is given explicitly in the instruction.

Example: Move 200 immediate ,R0
It places the value 200 in the register R0.The immediate mode used to specify the value of source operand. In assembly language, the immediate subscript is not appropriate so # symbol is used.
It can be re-written as

Move #200,R0

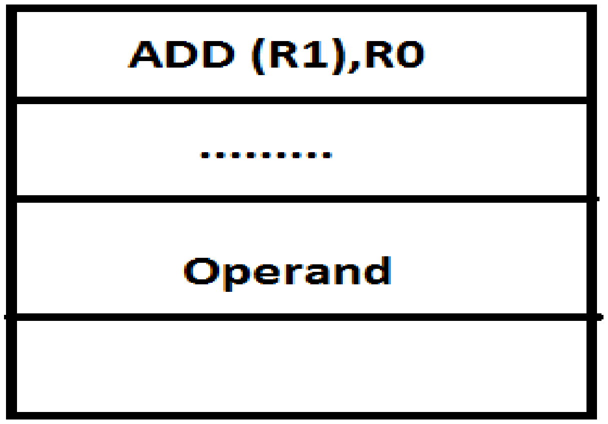
Assembly Syntax:	**Addressing Function**
Immediate #value	Operand =value

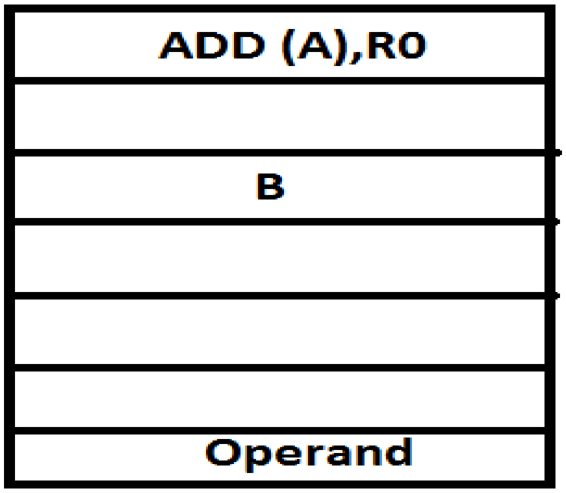
Indirection and Pointers

Instruction does not give the operand or its address explicitly. Instead it provides information from which the new address of the operand can be determined. This address is called effective Address(EA) of the operand.

Indirect Mode:
The effective address of the operand is the contents of a register .We denote the indirection by the name of the register or new address given in the instruction.

Fig:Indirect Mode





Address of an operand(B) is stored into R1 register.If we want this operand,we can get it through register R1(indirection).
The register or new location that contains the address of an operand is called the **pointer.**

Mode Assembler	**Syntax**	**Addressing Function**
Indirect	Ri , LOC	EA=[Ri] or EA=[LOC]

Indexing and Arrays:

Index Mode:

- The effective address of an operand is generated by adding a constant value to the contents of a register.
- The constant value uses either special purpose or general purpose register.
- We indicate the index mode symbolically as,

X(R_i)

Where **X** – denotes the constant value contained in the instruction
R_i – It is the name of the register involved.

The Effective Address of the operand is,
EA=X + [Ri]

Implied mode: In this mode,

- Operands are specified implicitly in the definition of the instruction.
- Example: The instruction complement accumulator
- Zero addressed instructions or stack organized computers are implied mode instructions.

Immediate Mode: In this mode,

1. An operand is specified in the instruction itself.

2. An immediate mode instruction has an operand field rather than an address field.
   Example: MOVE x,20
3. It is useful for initializing registers with constant value.
4. When this mode is used in register operation then it is known as "Register mode"

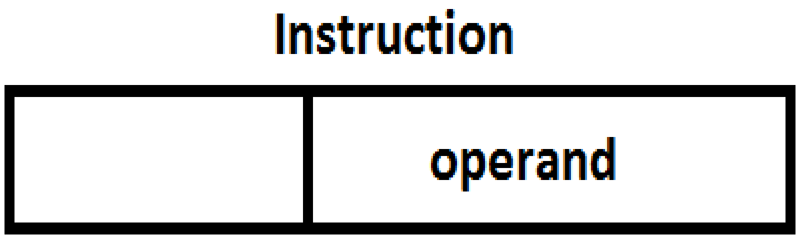
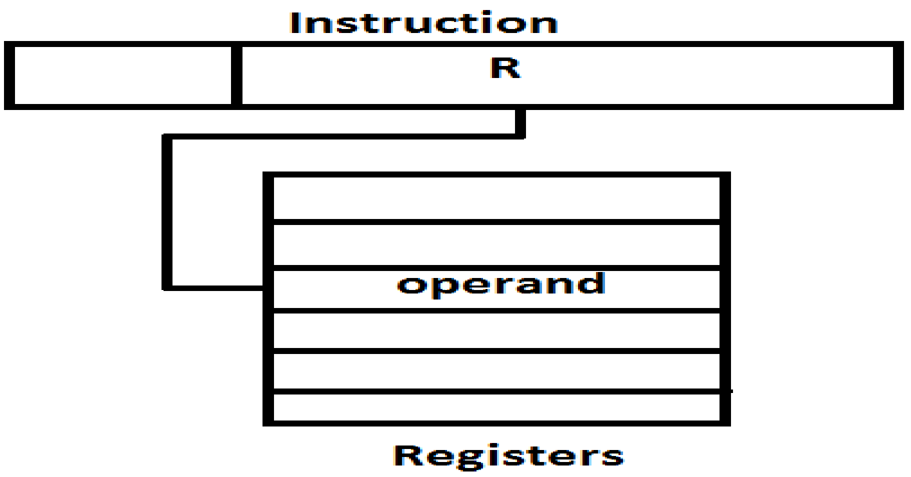


Fig:Immediate mode

Register Mode: In this mode,

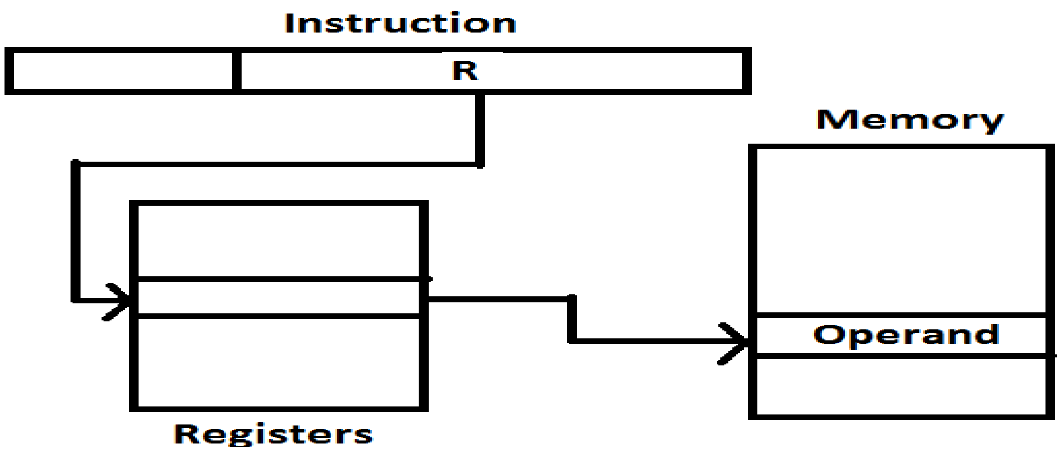
1. The operands in register that reside with in the cpu
2. The particular register is selected from a register field in the instruction.



Register Indirect Mode: In this mode,

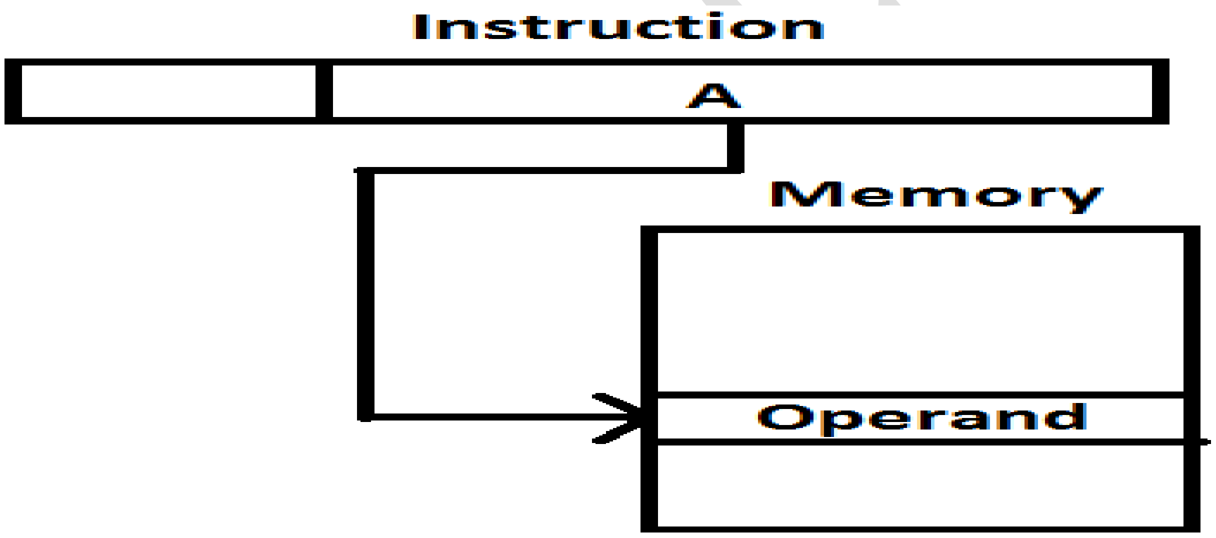
3. The instruction specifies a register in the CPU whose contents gives the address of the operand.
4. The selected register contains the address of the operand rather than the operand itself.

Advantage: The address field of the instruction uses fewer bits to select a register rather than memory.



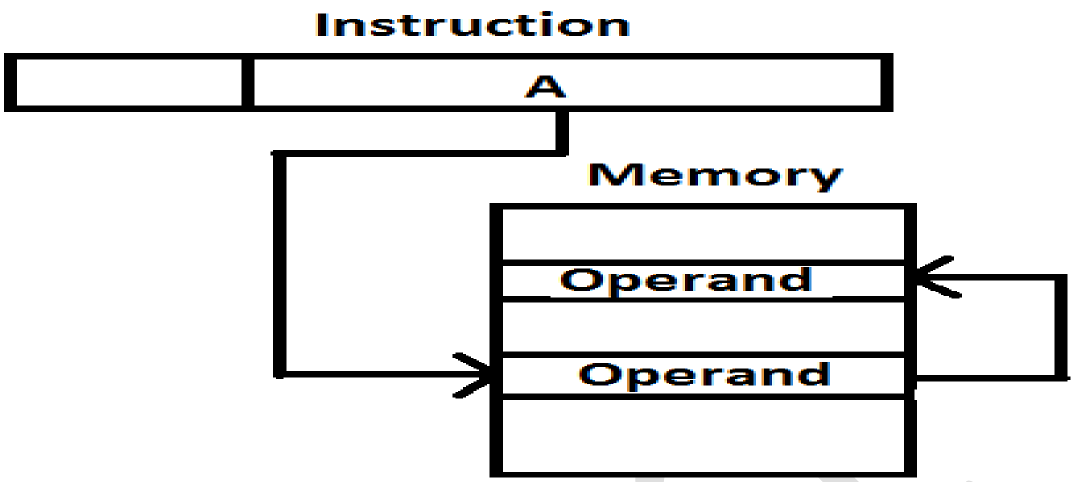
Direct Address Mode: In this mode,

1. The effective address is equal to the address part of the instruction. as shown below



In direct Address Mode: In this mode: In this mode

1. The address field of the instruction gives the address where the effective address is stored in memory.

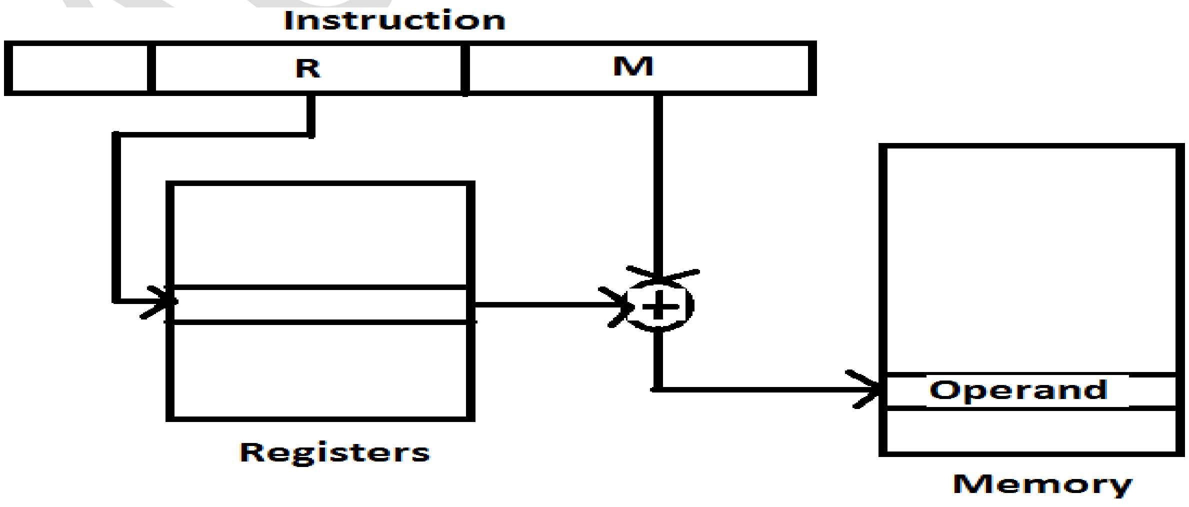


Relative Address mode: In this mode,
The contents of program counter is added to the address part of the instruction in order to obtain the effective address.
Effective Address(EA)=M[PC]+Address
Example: PC=824
Address=24
PC value after fetch Phase=824+1=825
EA=825+24=849

Displacement Mode: In This mode,

1. The content of base register is added to the address part of the instruction. It is used in a computer to facilitate the relocation of program in memory

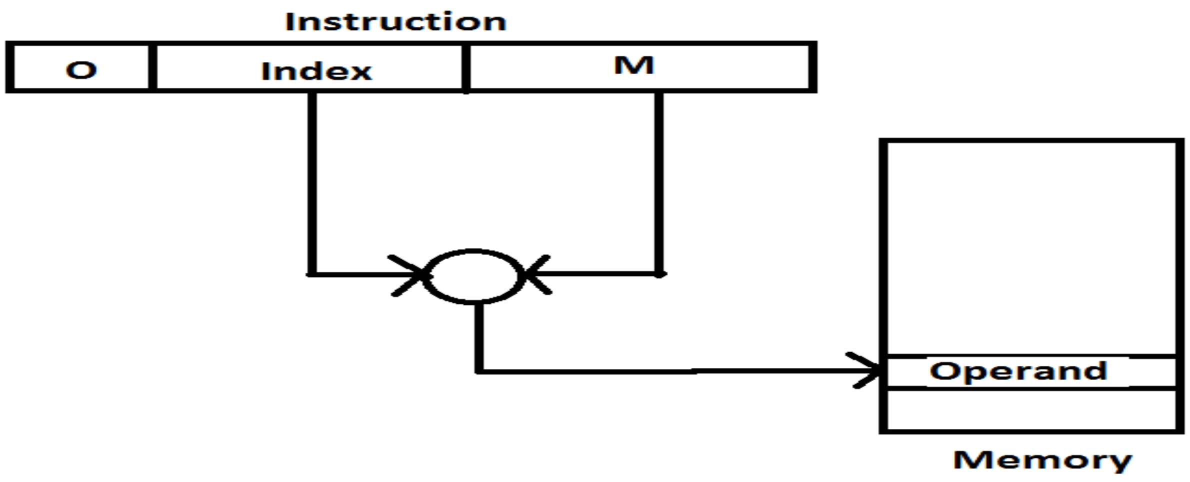
EA=M[Base Register]+Addres



Indexed Addressing Mode: In this mode,

2. The contents of index register is added to the to the address part of the instruction to obtain effective address.
3. The index register is a special CPU register that contain an other index value**.**

EA=Index register+Address field



Auto increment OR Decrement Mode: In this mode

When the address stored in the register refers to table of data in memory, it is necessary to increment or decrement register after every access to the table.

Effective Address= it is defined to be memory address obtained from the computation dictated by the given addressing mode.

It is the address where control branches in a response to a branch type instruction.

3.5 I/O OPERATIONS

ARM architecture uses memory mapped I/O .reading a character from a keyboard or sending a character to display can be done using program controlled I/O.
Suppose the bit 3 in each of the device status registers INSTATUS and OUTSTATUS contains the respective control flags SIN and SOUT. The DATAIN and DATAOUT registers are located at addresses INSTATUS+4 and OUTSTATUS+4, Immediately following status register locations.
READ:
Assume the address INSTATUS has been loaded into register R1. The instruction sequence

```
READWAIT     LDR   R3,[R1]
             TST   R3,#8
             BEQ   READWAIT
             LDRB  R3,[R1,#4]
```

Reads a character into register R3 when a key has been pressed on the keyboard. The TST operation performs logical and operation on its two operands and sets the condition code flags based on the result. The immediate operand 8 has single one in the bit3 position, therefore the result of the test operation will be zero if bit 3 of INSTATUS is zero and will be non-zero if bit 3 is one, signifying that a character is available in DATAIN.
The BEQ instruction branches back to READWAIT if the result is zero, looping until a key is pressed, which sets the bit 3 of the INSTATUS to one. Assuming that the OUTSTATUS has been loaded into register R2, the instruction sequence

```
READWAIT     LDR   R4,[R2]
             TST   R4,#8
             BEQ   READWAIT
             STRB  R3,[R2,#4]
```

Sends the character In register R3 to the dataout register when the display is ready to receive it.

```
READ     LDR   R3,[R1]      Load[INSTATUS]and
         TST   R3,#8        waits for character
         BEQ   READWAIT
         LDRB  R3,[R1,#4]   Read the character and
         STRB  R3,R0,#1     Store it in memory
ECHO     LDR   R4,[R2]      Load[OUTSTATUS]and
         TST   R4,#8        wait fot display
         BEQ   ECHO         to be ready
         STRB  R3,[R2,#4]   Send character to display
         TEQ   R3,#CR       if not carriage return
         BNE   READ         Read more characters
```

Fig:An ARM program that reads a line of characters and Display it