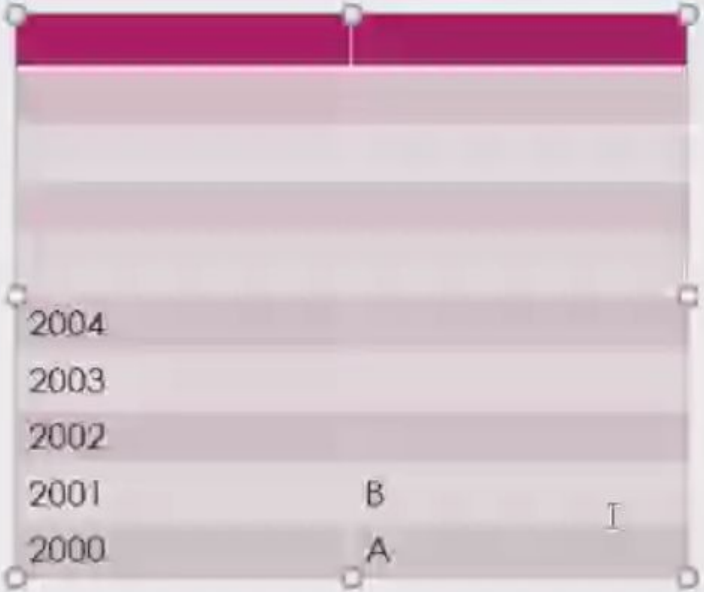
# Click to add title

- X=A*B+C*D
- LOAD A
- LOAD B
- MPY
- LOAD C
- LOAD D
- MPY
- ADD
- SRORE X
- HLT




Wadnido A. Obando

You

# Addressing Modes

Generating Memory Addresses

- How to specify the address of branch target?
- Can we give the memory operand address directly in a single Add instruction in the loop?
- Use a register to hold the address of NUM1; then increment by 4 on each pass through the loop.

You

You're presenting t

## ADDRESSING MODES

• The different ways in which the location of an operand is specified in an instruction are referred to as **Addressing Modes** (Table 2.1).

**Table 2.1** Generic addressing modes

| Name | Assembler syntax | Addressing function |
|---|---|---|
| Immediate | #Value | Operand = Value |
| Register | Ri | EA = Ri |
| Absolute (Direct) | LOC | EA = LOC |
| Indirect | (Ri)<br>(LOC) | EA = [Ri]<br>EA = [LOC] |
| Index | X(Ri) | EA = [Ri] + X |
| Base with index | (Ri,Rj) | EA = [Ri] + [Rj] |
| Base with index and offset | X(Ri,Rj) | EA = [Ri] + [Rj] + X |
| Relative | X(PC) | EA = [PC] + X |
| Autoincrement | (Ri)+ | EA = [Ri];<br>Increment Ri |
| Autodecrement | −(Ri) | Decrement Ri;<br>EA = [Ri] |

EA = effective address
Value = a signed number

## IMPLEMENTATION OF VARIABLE AND CONSTANTS

• **Variable** is represented by allocating a memory-location to hold its value.
• Thus, the value can be changed as needed using appropriate instructions.
• There are 2 accessing modes to access the variables:
    1) Register Mode
    2) Absolute Mode

**Register Mode**

• The operand is the co



You

| | | |
|---|---|---|
| Autoincrement | (Ri)+ | EA = [Ri]; Increment Ri |
| Autodecrement | −(Ri) | Decrement Ri; EA = [Ri] |

EA = effective address
Value = a signed number

**IMPLEMENTATION OF VARIABLE AND CONSTANTS**

- **Variable** is represented by allocating a memory-location to hold its value.
- Thus, the value can be changed as needed using appropriate instructions.
- There are 2 accessing modes to access the variables:
  1) Register Mode
  2) Absolute Mode

**Register Mode**

- The operand is the contents of a register.
- The name (or address) of the register is given in the instruction.
- Registers are used as temporary storage locations where the data in a register are accessed.
- For example, the instruction

  *Move R1, R2* ;Copy content of register R1 into register R2.

**Absolute (Direct) Mode**

- The operand is in a memory-location.
- The address of memory-location is given explicitly in the instruction.
- The absolute mode can represent global variables in the program.
- For example, the instruction

  *Move LOC, R2* ;Copy content of memory-location LOC into register R2.

**Immediate Mode**

- The operand is given explicitly in the instruction.
- For example, the instruction

  *Move #200, R0* ;Place the value 200 in register R0.
- Clearly, the immediate mode is only used to specify the value of a source-operand.

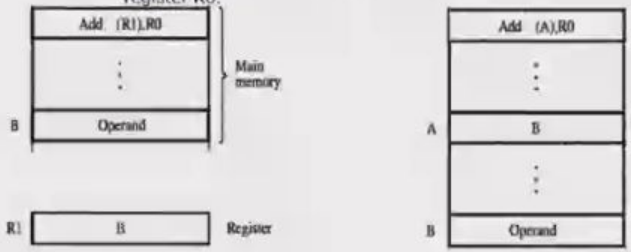
**INDIRECTION AND POINTERS**

- Instruction does not give the operand or its address explicitly.
- Instead, the instruction 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 EA of the operand is the contents of a register(or memory-location).
- The register (or memory-location) that contains the address of an operand is called a **Pointer**.
- We denote the indirection by
  - → name of the register or
  - → new address given in the instruction.

E.g: *Add (R1),R0* ;The operand is in memory. Register R1 gives the effective-address (B) of the operand. The data is read from location B and added to contents of register R0.




(a) Through a general-purpose register (b) Through a memory location

**Figure 2.11** Indirect addressing.

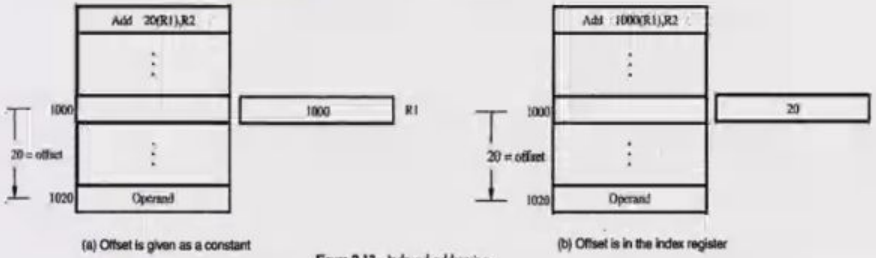
- To execute the Add instruction in fig 2.11 (a), the processor uses the value which is in register R1, as the EA of the operand.
- It requests a read operation from the memory to read the contents of location B. The value read is the desired operand, wh
- Indirect addressing th 2.11(b). In this case,

## INDEXING AND ARRAYS

- A different kind of flexibility for accessing operands is useful in dealing with lists and arrays.

**Index mode**

- The operation is indicated as X(Ri)
  where X=the constant value which defines an offset(also called a displacement).
  Ri=the name of the index register which contains address of a new location.
- The effective-address of the operand is given by EA=X+[Ri]
- The contents of the index-register are not changed in the process of generating the effective-address.
- The constant X may be given either
  → as an explicit number or
  → as a symbolic-name representing a numerical value.




(a) Offset is given as a constant (b) Offset is in the index register

**Figure 2.13** Indexed addressing.

- Fig(a) illustrates two ways of using the Index mode. In fig(a), the index register, R1, contains the address of a memory-location, and the value X defines an offset(also called a displacement) from this address to the location where the operand is found.
- To find EA of operand:
  Eg: Add 20(R1), R2
  EA=>1000+20=1020
- An alternative use is illustrated in fig(b). Here, the constant X corresponds to a memory address, and the contents of the index register define the offset to the operand. In either case, the effective-address is the sum of two values; one is given explicitly in the instruction, and the other is stored in a register.

#LIST,R0

**Base with Index Mode**

- Another version of the Index mode uses 2 registers which can be denoted as
  (Ri, Rj)
- Here, a second register may be used to contain the offset X.
- The second register is usually called the *base register*.
- The effective-address of the operand is given by EA=[Ri]+[Rj]
- This form of indexed addressing provides more flexibility in accessing operands because
  both components of the effective-address can be changed.

**Base with Index & Offset Mode**

- Another version of the Index mode uses 2 registers plus a constant, which can be denoted as
  X(Ri, Rj)
- The effective-address of the operand is given by EA=X+[Ri]+[Rj]
- This added flexibility is useful in accessing multiple components inside each item in a record, where the beginning of an item is specified by the (Ri, Rj) part of the addressing-mode. In other words, this mode implements a 3-dimensional array.

**RELATIVE MODE**

- This is similar to index-mode with one difference:
  The effective-address is determined using the PC in place of the general purpose regis
- The operation is indicated as X(PC)

## Base with Index Mode

- Another version of the Index mode uses 2 registers which can be denoted as
  (Ri, Rj)
- Here, a second register may be used to contain the offset X.
- The second register is usually called the *base register*.
- The effective-address of the operand is given by EA=[Ri]+[Rj]
- This form of indexed addressing provides more flexibility in accessing operands because both components of the effective-address can be changed.

| | | |
|---|---|---|
| | ADD<br>(R1,R0),R2 | |
| | .. | |
| | .. | |
| | .. | |
| 1000 | | |
| | .. | |
| | | |
| | | |

- The effective-address of the operand is given by EA=X+[Ri]+[Rj]
- This added flexibility is useful in accessing multiple components inside each item in a record, where the beginning of an item is specified by the (Ri, Rj) part of the addressing-mode. In other words, this mode implements a 3-dimensional array.

**RELATIVE MODE**

- This is similar to index-mode with one difference:
  The effective-address is determined using the PC in place of the general purpose register Ri.
- The operation is indicated as X(PC).
- X(PC) denotes an effective-address of the operand which is X locations above or below the current contents of PC.
- Since the addressed-location is identified "relative" to the PC, the name Relative mode is associated with this type of addressing.
- This mode is used commonly in conditional branch instructions.
- An instruction such as

  *Branch > 0 LOOP* ;Causes program execution to go to the branch target location identified by name LOOP if branch condition is satisfied.

**ADDITIONAL ADDRESSING MODES**

**1) Auto Increment Mode**

Effective-address of operand is c... the instruction (Fig:

# Click to add title

- ADD (R2)+,R0

| R2 | 1040 |
|---|---|
| | |
| | |
| | |

| R0 | 10 |
|---|---|
| | |
| | |
| | |

| 103C | |
|---|---|
| 1040 | **10** |
| 1044 | 20 |
| 1048 | |

| | |
|---|---|
| R2 | 1044 |
| R0 | 20H |
| | |



You