CO: Computer Organization

Addressing Modes

Indian Institute of Information Technology, Sri City

Jan - May - 2018

http://co-iiits.blogspot.in/

Addressing Modes

Location of an operand in an instruction.

1. Immediate Addressing Mode: Load $R_0, \#100$
   (Operand is given explicitly in the instruction)

2. Absolute(Direct) Addressing Mode: Load R_0, A
   (Operand is in a memory location and it is given explicitly in the instruction)

3. Register Addressing Mode: Add R_0, R_1
   (Operand is in the contents of a processor register and the name of the register is given explicitly in the instruction)

Write an assembly code for the following HLL statements:

A=6+B

C=A+D

Addressing Modes

Location of an operand in an instruction.

1. Immediate Addressing Mode: Load $R_0, \#100$
   (Operand is given explicitly in the instruction)

2. Absolute(Direct) Addressing Mode: Load R_0, A
   (Operand is in a memory location and it is given explicitly in the instruction)

3. Register Addressing Mode: Add R_0, R_1
   (Operand is in the contents of a processor register and the name of the register is given explicitly in the instruction)

Write an assembly code for the following HLL statements:

A=6+B

C=A+D

Addressing Modes

Location of an operand in an instruction.

1. Immediate Addressing Mode: Load $R_0, \#100$
   (Operand is given explicitly in the instruction)

2. Absolute(Direct) Addressing Mode: Load R_0, A
   (Operand is in a memory location and it is given explicitly in the instruction)

3. Register Addressing Mode: Add R_0, R_1
   (Operand is in the contents of a processor register and the name of the register is given explicitly in the instruction)

Write an assembly code for the following HLL statements:

A=6+B

C=A+D

Addressing Modes

Location of an operand in an instruction.

1. Immediate Addressing Mode: Load $R_0, \#100$
   (Operand is given explicitly in the instruction)

2. Absolute(Direct) Addressing Mode: Load R_0, A
   (Operand is in a memory location and it is given explicitly in the instruction)

3. Register Addressing Mode: Add R_0, R_1
   (Operand is in the contents of a processor register and the name of the register is given explicitly in the instruction)

Write an assembly code for the following HLL statements:

A=6+B

C=A+D

Addressing Modes

Location of an operand in an instruction.

1. Immediate Addressing Mode: Load $R_0, \#100$
   (Operand is given explicitly in the instruction)

2. Absolute(Direct) Addressing Mode: Load R_0, A
   (Operand is in a memory location and it is given explicitly in the instruction)

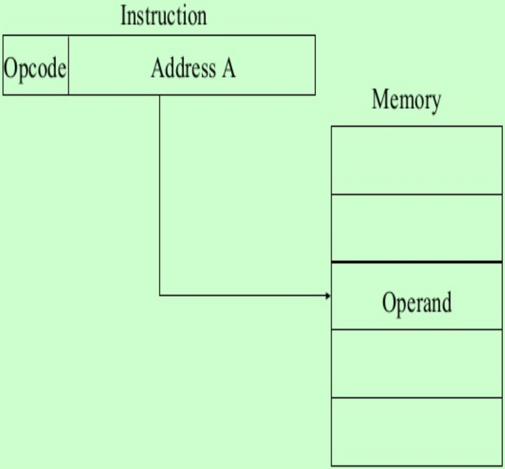
3. Register Addressing Mode: Add R_0, R_1
   (Operand is in the contents of a processor register and the name of the register is given explicitly in the instruction)

Write an assembly code for the following HLL statements:

A=6+B

C=A+D

Direct Addressing Diagram



Addressing Modes

Location of an operand in an instruction.

Assembly code for the following HLL statements:

A=6+B

C=A+D

- Load $R_0, \#6 - - - -- >$ Immediate
- Load $R_1, B - - - - - - >$ Absolute
- Add $R_0, R_1 - - - -- >$ Register
- Load $R_2, D - - - -- >$ Absolute
- Add $R_0, R_2 - - - -- >$ Register
- Store $C, R_0 - - - -- >$ Absolute

Addressing Modes

Indirect Address Mode: The effective address of an operand is either the contents of a register or memory location.
Example: $*A = *B + *C$

Load $R_0, (B)$**;**
Load $R_1, (C)$**;**
Add R_0, R_1**;**
Store $(A), R_0$**;**

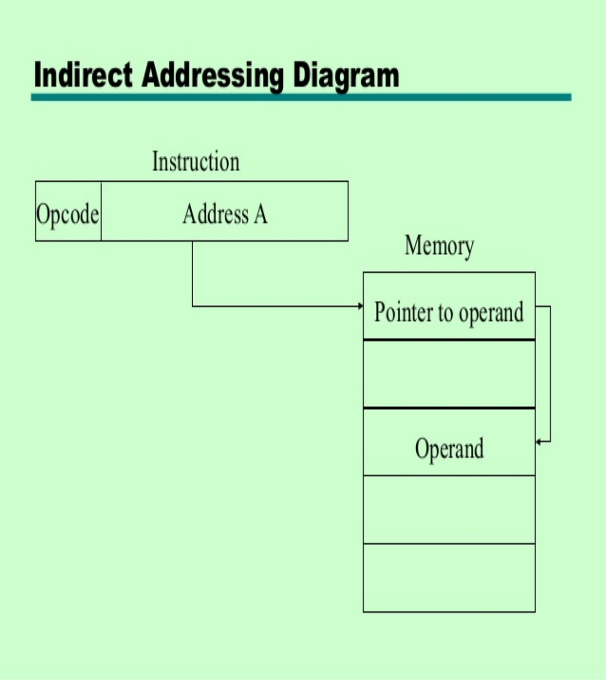
Load R_0, B**;**
Load R_1, C**;**
Add $R_2, (R_0), (R_1)$**;**
Load R_3, A**;**
Store $(R_3), R_2$**;**

Addressing Modes

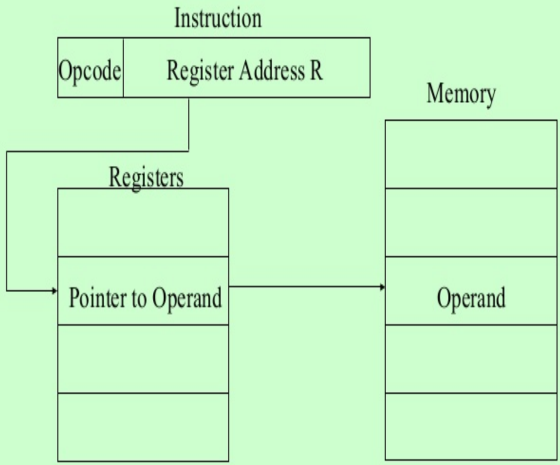
Indirect Address Mode: The effective address of an operand is either the contents of a register or memory location.
Example: $*A = *B + *C$

Load $R_0, (B)$**;**
Load $R_1, (C)$**;**
Add R_0, R_1**;**
Store $(A), R_0$**;**

Load R_0, B**;**
Load R_1, C**;**
Add $R_2, (R_0), (R_1)$**;**
Load R_3, A**;**
Store $(R_3), R_2$**;**


Indirect Addressing Diagram
Instruction
Opcode
Address A
Memory
Pointer to operand
Operand

Register Indirect Addressing Diagram



Addressing Modes

Index Addressing Mode: The effective address(EA) of the operand is generated by adding a constant value to the contents of a register.

$for(i = 0; i < N; i++)$

$S = S + A[i]$

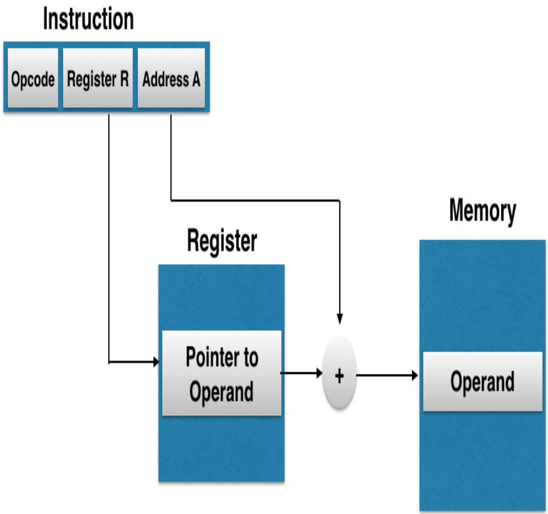
	Load R_0, A;	
	Load $R_1, \#N$;	
	Load $R_2, \#0$;	
LOOP	**Add** $R_2, 0(R_0)$;	$EA = 0 + [R_0]$
	Add $R_0, \#4$;	
	Decrement R_1;	
	Branch> 0 **LOOP**;	
	Store S, R_2;	

Addressing Modes

Index Addressing Mode: The effective address(EA) of the operand is generated by adding a constant value to the contents of a register.
$for(i = 0; i < N; i++)$
$S = S + A[i]$

	Load R_0, A**;**	
	Load $R_1, \#N$**;**	
	Load $R_2, \#0$**;**	
LOOP	**Add** $R_2, 0(R_0)$**;**	$EA = 0 + [R_0]$
	Add $R_0, \#4$**;**	
	Decrement R_1**;**	
	Branch> 0 **LOOP;**	
	Store S, R_2**;**	

Index Addressing Mode



Addressing Modes

Relative Addressing Mode: The effective address(EA) of the operand is generated by adding a constant value to the contents of Program Counter(PC).

$for(i = 0; i < N; i + +)$
$S = S + A[i]$

1000 **Load** R_0, A**;**
1004 **Load** $R_1, \#N$**;**
1008 **Load** $R_2, \#0$**;**
100C LOOP **Add** $R_2, 0(R_0)$**;** $EA = 0 + [R_0]$
1010 **Add** $R_0, \#4$**;**
1014 **Decrement** R_1**;**
1018 **Branch>0 LOOP; On success PC=-16+PC**
101C **Store** S, R_2**;**

http://rextester.com/FUMK95551

Addressing Modes

Relative Addressing Mode: The effective address(EA) of the operand is generated by adding a constant value to the contents of Program Counter(PC).

$for(i = 0; i < N; i++)$

$S = S + A[i]$

1000 **Load** R_0, A**;**

1004 **Load** $R_1, \#N$**;**

1008 **Load** $R_2, \#0$**;**

100C LOOP **Add** $R_2, 0(R_0)$**;** $EA = 0 + [R_0]$

1010 **Add** $R_0, \#4$**;**

1014 **Decrement** R_1**;**

1018 **Branch>0 LOOP; On success PC=-16+PC**

101C **Store** S, R_2**;**

http://rextester.com/FUMK95551

Addressing Modes

Auto-Increment Addressing Mode: The effective address(EA) of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of the register points to the next data item.

Load R_0, A**;**
Load $R_1, \#N$**;**
Load $R_2, \#0$**;**
LOOP **Add** $R_2, (R_0)+$**;** $EA = [R_0]$ **and** $R_0 = R_0 + 4$
Decrement R_1**;**
Branch>0 LOOP; On success PC=-12+PC
Store S, R_2**;**

Addressing Modes

Auto-Decrement Addressing Mode: The contents of a register specified in the instruction are first automatically decremented and then the EA of operand is the contents of the register.

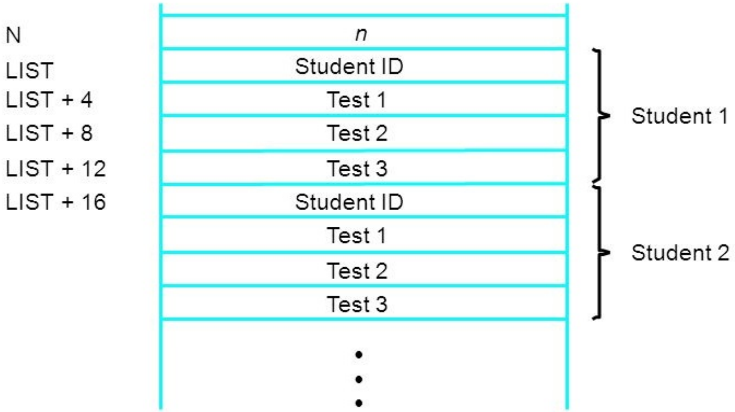
Add $R_2, -(R_0)$**;** $\quad$ $R_0 = R_0 - 4$ **and** $EA = [R_0]$

Addressing Modes

Name	Syntax	Addressing Function
Immediate	#Value	Operand = Value
Register	Ri	EA = Ri
Absolute (Direct)	LOC	EA = LOC
Indirect	(Ri) (LOC)	EA = [Ri] EA = [LOC]
Index	X(Ri)	EA = [Ri] + X
Relative	X(PC)	EA = [PC] + X
Autoincrement	(Ri)+	EA = [Ri]; Ri ← [Ri] + 1
Autodecrement	±(Ri)	Ri ← [Ri] ± 1; EA = [Ri]

EA = Effective Address

There are n students in a class. Information of n students is stored using the concept of records. Each record consists of a student's ID, followed by marks on three tests. Memory organization is shown in the below diagram. Write an assembly code to compute the sum of all scores obtained on each of the tests.



	Load $R_0, LIST$**;**	
	Sub R_1, R_1**;**	
	Sub R_2, R_2**;**	
	Sub R_3, R_3**;**	
	Load R_4, N**;**	
LOOP	**Add** $R_1, 4(R_0)$**;**	$EA = 4 + [R_0]$
	Add $R_2, 8(R_0)$**;**	$EA = 8 + [R_0]$
	Add $R_3, 12(R_0)$**;**	$EA = 12 + [R_0]$
	Add $R_0, \#16$**;**	
	Decrement R_4**;**	
	Branch>0 **LOOP;**	
	Store $Sum1, R_1$**;**	
	Store $Sum2, R_2$**;**	
	Store $Sum3, R_3$**;**	