Instruction Set Architecture

Instruction Format

The instruction is divided into two fields

- Opcode field
- Operand field

This operand field further divided into one to four fields.

Simple instruction format

Opcode	Operand Address1	Operand Address2	Result Address1	Next Instruction

Instruction Set is categorized into types based on,

- Operation performed
- Number of operand addresses
- Addressing modes

Based on Operation

- Data Movement

  Memory : LOAD, STORE, MOV

- Data Processing

  Arithmetic : Add, Sub, MUL

- Control Instructions

  Conditional : JNZ, JZ

  Un Conditional: Jump

- I\O Instructions: IN, OUT
- Logic Instructions: AND, OR

Based on number of **operand address** in the instruction.

4 Address Instruction
3 Address Instruction
2 Address Instruction
1 Address Instruction
0 Address Instruction

For a two-operand arithmetic instruction, five items need to be specified

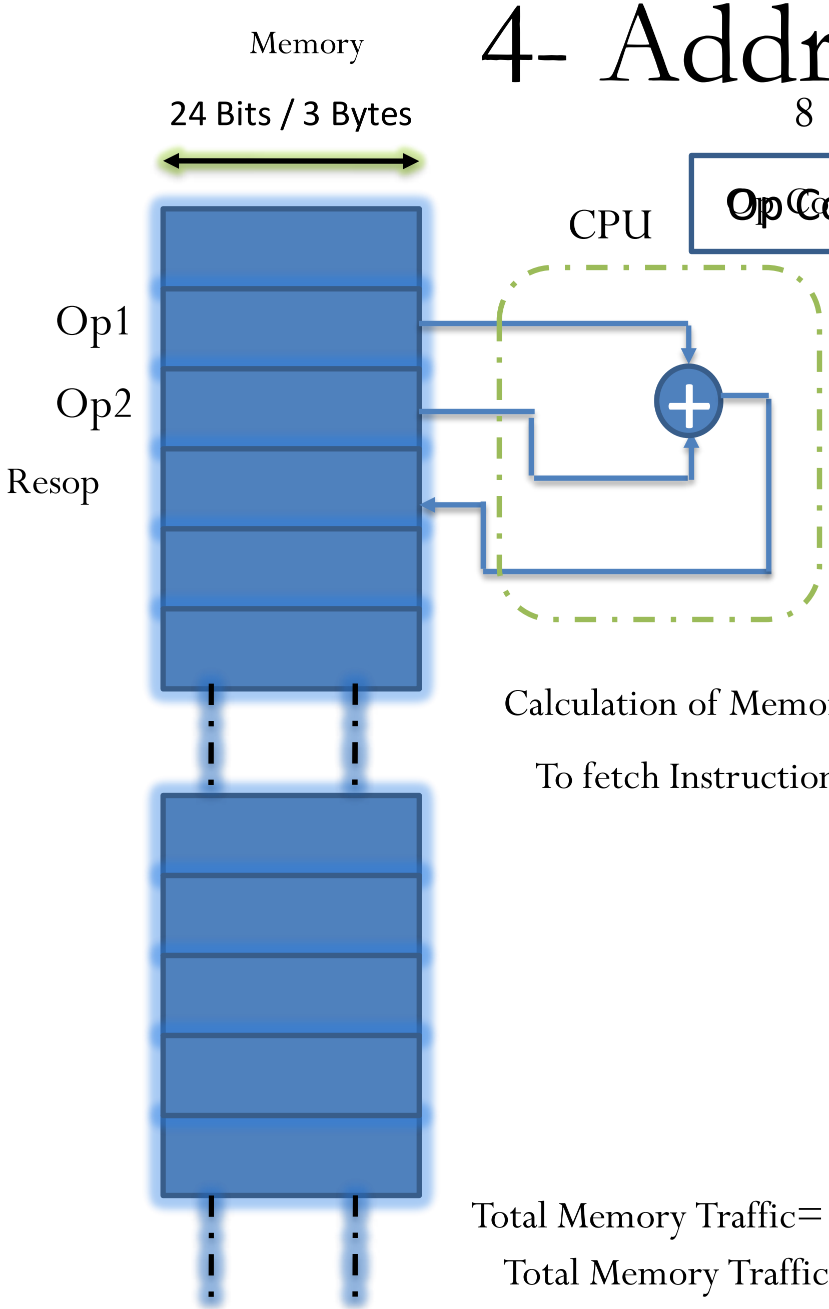
1. Operation to be performed (opcode)
2. Location of the first operand
3. Location of the second operand
4. Place to store the result
5. Location of next instruction to be executed

Assumptions

24-bit memory address (3 bytes)

128 instructions (7 bits rounded to 1 byte)

4- Address Instruction



8	24	24	24	24
Op Code	ResAddr	Op1Addr	Op2Addr	NextiAddr

Example: add M1,M2,M3, nexti

M(1)←M(2)+M(3)

Memory Required to store an Instruction:

5 x 3 bytes = 15 Bytes

Memory Required to Encode an Instruction:

1Byte+ 4 x 3 bytes = 13 Bytes

Calculation of Memory Accesses

To fetch Instruction itself

Opcode=1
Op1Addr=1
Op2Addr=1
ResAddr=1
NextiAddr=1

Total=5

To Execute an Instruction

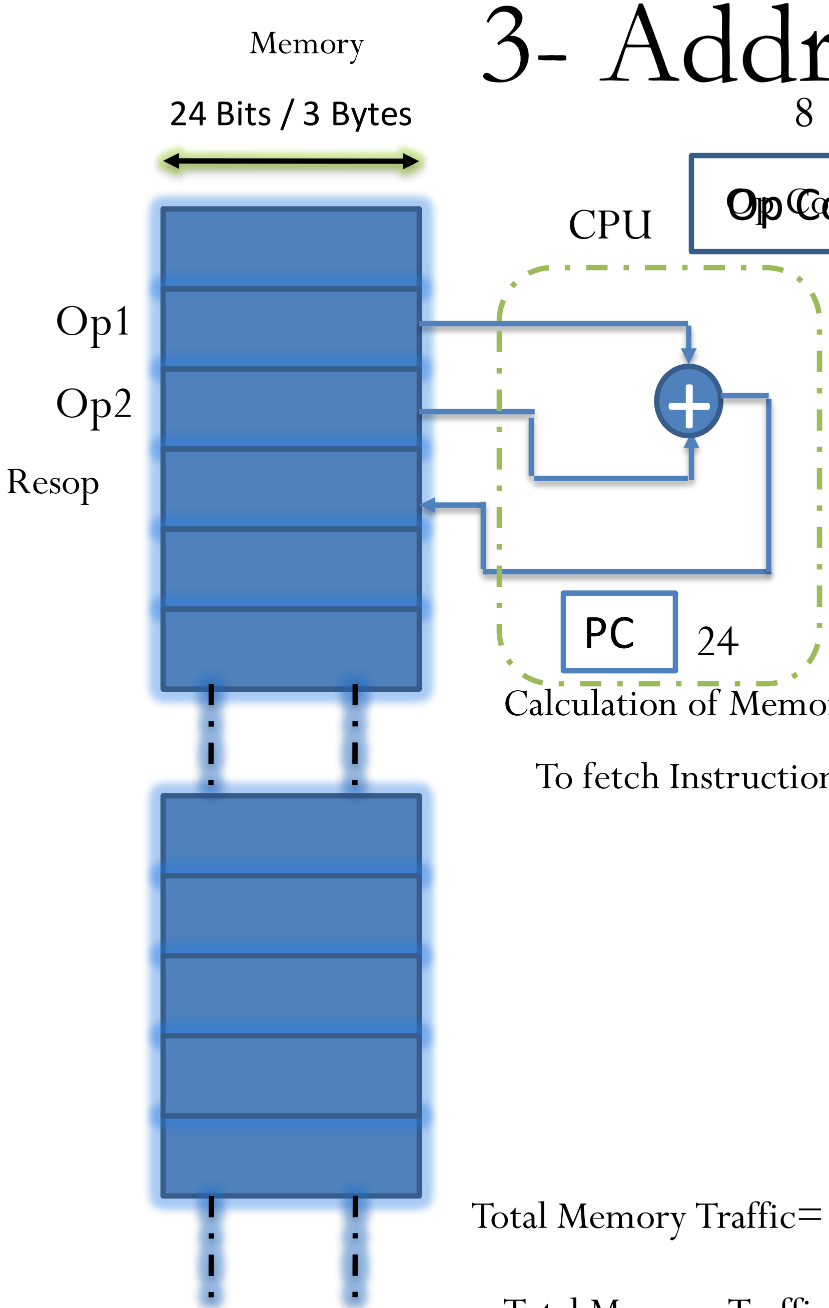
Op1=1
Op2=1
Res=1

Total=3

Total Memory Traffic= No. of M/A to fetch + No. of M/A to Execute

Total Memory Traffic= 5 + 3 =8

3- Address Instruction



8	24	24	24
Op Code	ResAddr	Op1Addr	Op2Addr

Example: add M1,M2,M3

M(1)←M(2)+M(3)

Memory Required to store an Instruction:

4 x 3 bytes = 12 Bytes

Memory Required to Encode an Instruction:

1Byte+ 3 x 3 bytes = 10 Bytes

Calculation of Memory Accesses

To fetch Instruction itself

Opcode=1
Op1Addr=1
Op2Addr=1
ResAddr=1

Total=4

To Execute an Instruction

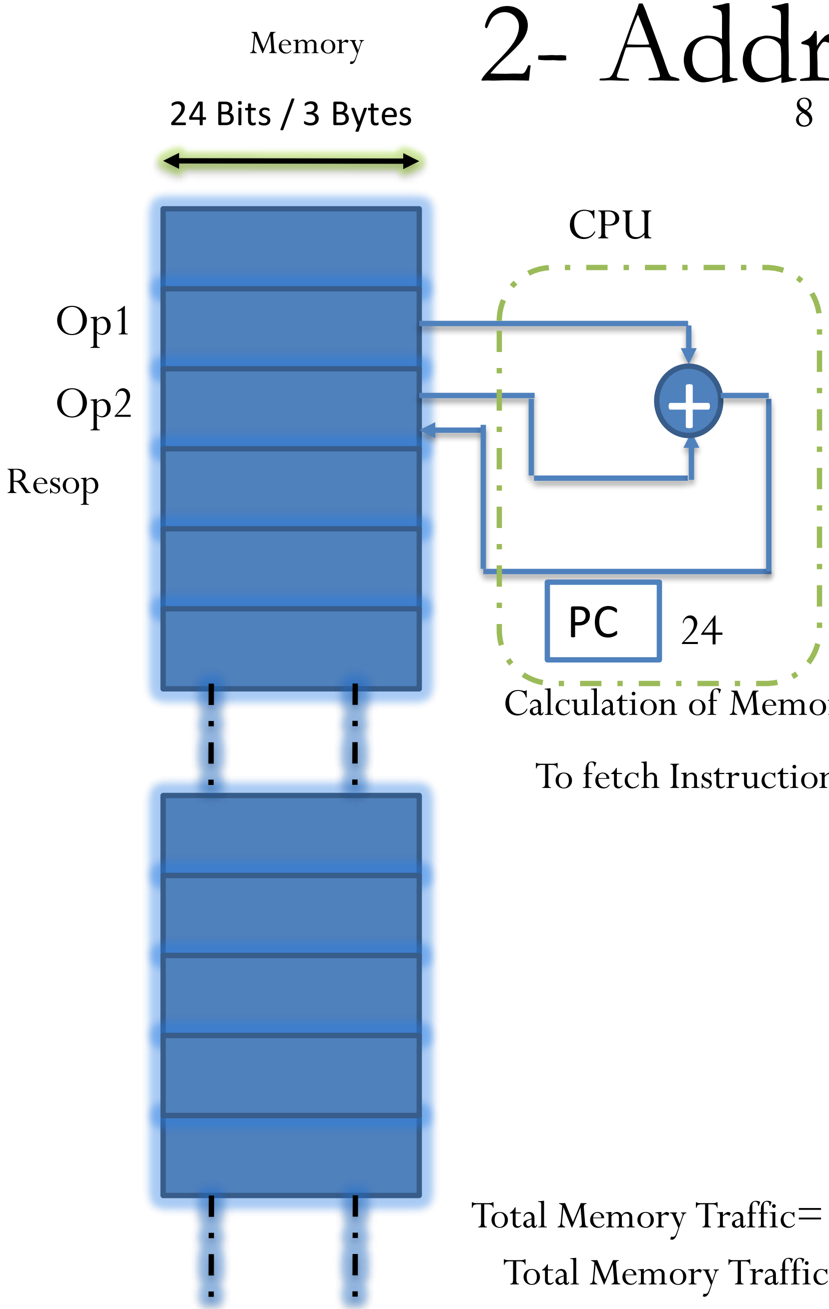
Op1=1
Op2=1
Res=1

Total=3

Total Memory Traffic= No. of M/A to fetch + No. of M/A to Execute

Total Memory Traffic= 4+ 3 =7

2- Address Instruction



8

24	24	24
Op Code	Op1Addr	Op2Addr

Example: add M2,M3

M(2)←M(2)+M(3)

Memory Required to store an Instruction:

3 x 3 bytes = 09 Bytes

Memory Required to Encode an Instruction:

1Byte+ 2 x 3 bytes = 7 Bytes

Calculation of Memory Accesses

To fetch Instruction itself

Opcode=1
Op1Addr=1
Op2Addr=1

Total=3

To Execute an Instruction

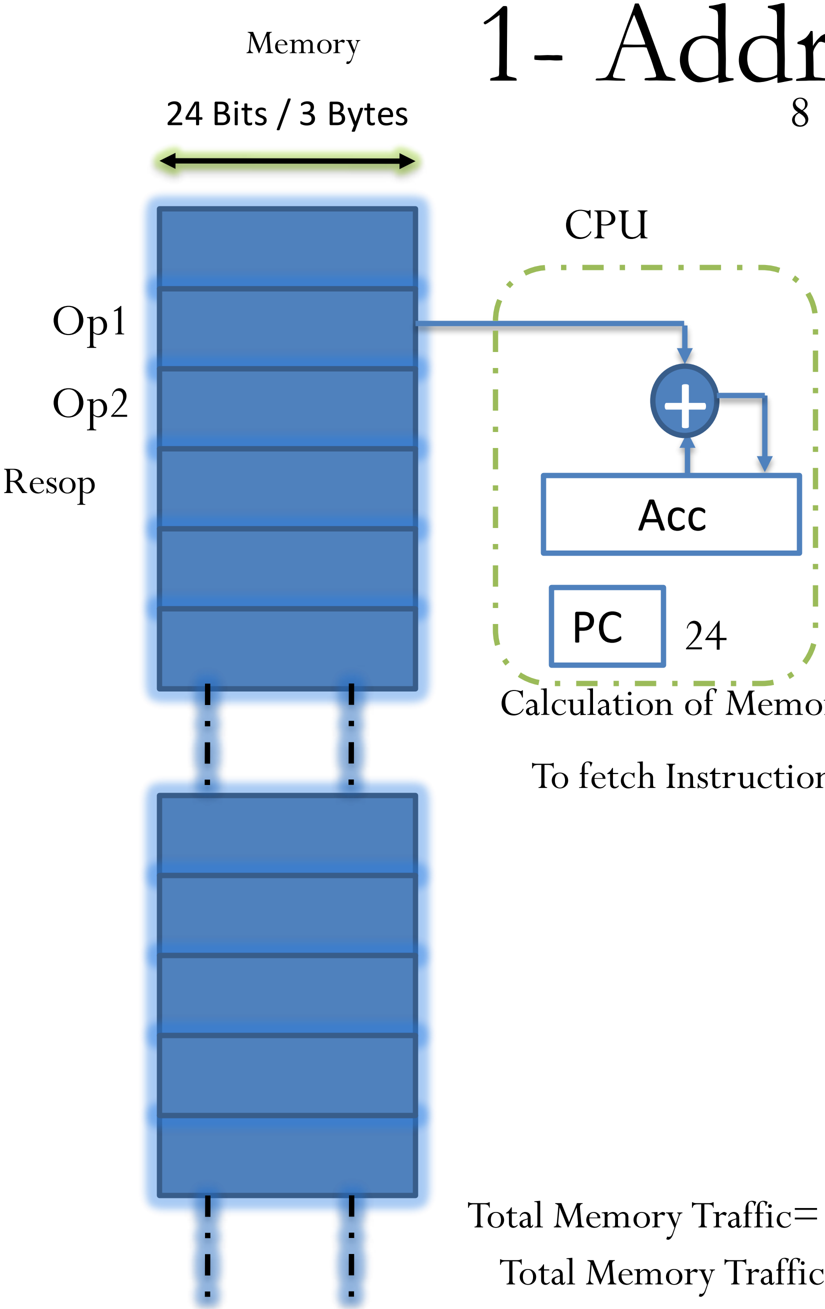
Op1=1
Op2=1
Res=1

Total=3

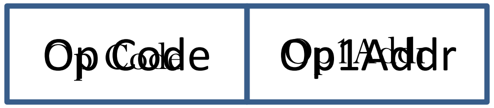
Total Memory Traffic= No. of M/A to fetch + No. of M/A to Execute

Total Memory Traffic= 3+ 3 =6

1- Address Instruction



8 24 24



Example: add M2

Memory Required to store an Instruction:

2 x 3 bytes = 06 Bytes

Memory Required to Encode an Instruction:

1Byte+ 1 x 3 bytes = 4Bytes

Calculation of Memory Accesses

To fetch Instruction itself

Opcode=1
Op1Addr=1

Total=2

To Execute an Instruction

Op1=1

Total=1

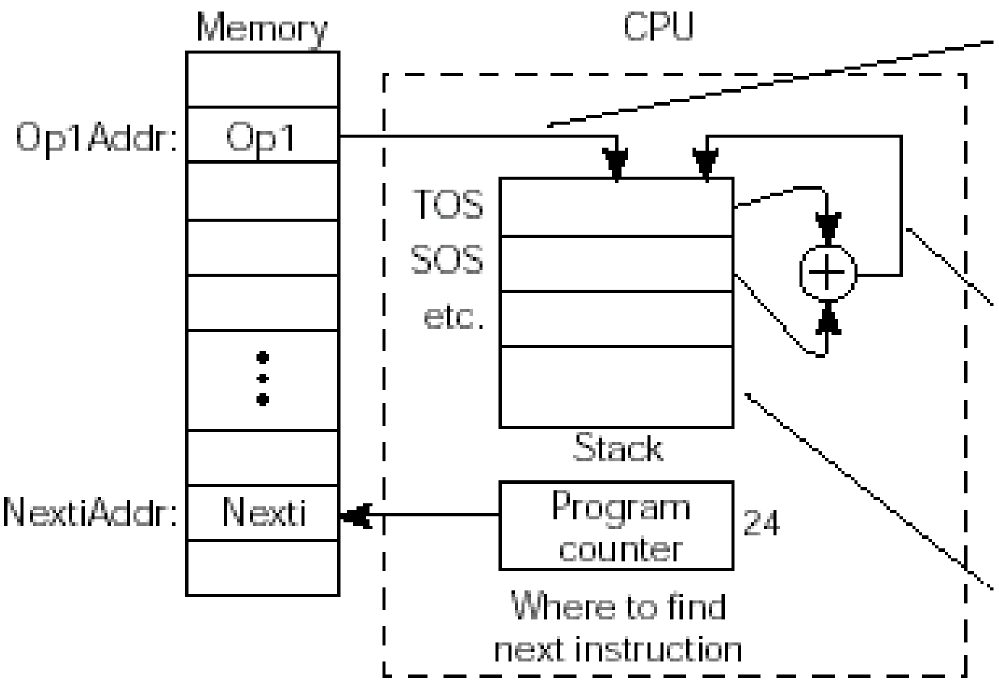
Total Memory Traffic= No. of M/A to fetch + No. of M/A to Execute

Total Memory Traffic= 2+ 1 =3

0-Address Instruction

- Uses a push down stack in CPU
- Arithmetic uses stack for both operands and the result

Computer must have a 1-address instruction to push and pop operands to and from the stack



Push Op1(TOS ← Op1)

Push	Op1Addr

Add (TOS ← TOS + SOS)

add

Where to find operands and where to put the result (on the Stack)

Comparisons

Instruction Type	Memory To Store in Bytes	Memory To Encode in Bytes	M/As to fetch an Instruction	M/As to Execute an Instruction	Memory Traffic
4-address	5 x 3 = 15	1+(4 x 3) = 13	5	3	5+3=8
3-Address	4 x 3 = 12	1+(3 x 3) = 10	4	3	4+3=7
2-Address	3 x 3 = 09	1+(2 x 3) = 07	3	3	3+3=6
1-Address	2 x 3 = 06	1+(1 x 3) = 04	2	1	2+1=3
0-Address	1 x 3 = 03	1+(0 x 3) = 01	1	0	1+0=1

Evaluate a = (b+c)*d - e

3-Address

add a, b, c	a←b+c
mpy a, a, d	a←a*d
sub a, a, e	a←a-e

2-Address

load a, b	a←b
add a, c	a←a+c
mpy a, d	a←a*d
sub a, e	a←a-e

1-Address

load b	Acc←b
add c	Acc←Acc+c
mpy d	Acc←Acc*d
sub e	Acc←Acc-e
store a	a←Acc

0-Address

push b

push c

add

push d

mpy

push e

sub

pop a

		Memory to Store	Memory to encode	M/As to Fetch	M/As to Execute	Memory Traffic
add a, b, c	a←b+c	4*3=12	1+(3*3)=10	4	3	4+3=7
mpy a, a, d	a←a*d	4*3=12	1+(3*3)=10	4	3	4+3=7
sub a, a, e	a←a-e	4*3=12	1+(3*3)=10	4	3	4+3=7
		36	**30**	**12**	**9**	**21**

		Memory to Store	Memory to encode	M/As to Fetch	M/As to Execute	Memory Traffic
load a, b	a←b	3*3=9	1+(2*3)=7	3	2	3+2=5
add a, c	a←a+c	3*3=9	1+(2*3)=7	3	3	3+3=6
mpy a, d	a←a*d	3*3=9	1+(2*3)=7	3	3	3+3=6
sub a, e	a←a-e	3*3=9	1+(2*3)=7	3	3	3+3=6
		36	**28**	**12**	**11**	**23**

		Memory to Store	Memory to encode	M/As to Fetch	M/As to Execute	Memory Traffic
load b	Acc←b	2*3=6	1+(1*3)=4	2	1	2+1=3
add c	Acc←Acc+c	2*3=6	1+(1*3)=4	2	1	2+1=3
mpy d	Acc←Acc*d	2*3=6	1+(1*3)=4	2	1	2+1=3
sub e	Acc←Acc-e	2*3=6	1+(1*3)=4	2	1	2+1=3
store a	a←Acc	2*3=6	1+(1*3)=4	2	1	2+1=3
		30	**20**	**10**	**5**	**15**

push b	6	4	2	1	3
push c					
add	3	1	1	0	1
push d					
mpy					
push e					
sub					
pop a					
	39	**23**	**13**	**5**	**18**

- **Assume,**
  - **Size of memory address is 2bytes**
  - **Size of operant is 2bytes**
  - **Size of a memory location is 1byte**
  - **Size of opcode is 1byte**

Then,

- Evaluate X=(A+B) * (C+D)
- Evaluate X=(a/b+c*d)/(d*e-f+c/a)+g
- Evaluate Y=(A-B)/[C+(D/E)]

Evaluate a = (b+c)*d - e

3-Address

add x, a, b

Add c, c, d

Mul x, x, c

2-Address

Add a, b

add c, d

Mul a, c

Load x, a

1-Address

load a

add b

store x

Load c

Add d
Mul x
Store x

0-Address

push a
Push b
Add
Push c
Push d
Mul
Pop x

Evaluate a = (b+c)*d - e

3-Address

```
add x, a, b    7+6
Add c, c, d
Mul x, x, c
```

2-Address

```
Add  a, b     5+6
add  c, d
Mul  a, c
Load x, a      5+4
```

1-Address

```
load a      3+2
add b
store x
Load c
Add d
Mul x
Store x
```

0-Address

```
push a
Push b
Add
Push c
Push d
Mul
Pop x
```

References

Reference Book

- W. Stallings, Computer organization and architecture, Prentice-Hall,2000
- J. P. Hayes, Computer system architecture, McGraw Hill,2000
- Vincent .P. Heuring, Harry F. Jordan " Computer System design and Architecture" Pearson, 2nd Edition, 2003