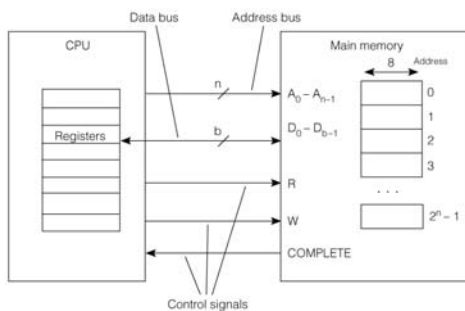


## Class 3: Instruction Set Architectures

## ISA Components

- Storage cells
  - Registers, memory, etc.
- The Machine Instruction Set
  - Set of possible operations
- The Instruction Format
  - Size and meaning of fields within the instruction

## Memory

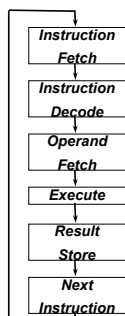


Copyright © 2004 Pearson Prentice Hall, Inc.

## What do instructions need to specify?

- Which operation to perform
  - `add r0, r1, r3` // operation code
- Where to find the operand or operands
  - `add r0, r1, r3` // source operands
- Place to store result
  - `add r0, r1, r3` // destination
- Location of next instruction
  - `add r0, r1, r3`  
`br endloop`

## Basic Instruction Cycle



## Classes of Instructions

- **Data movement instructions**
  - **Load**—source is memory and destination is register
  - **Store**—source is register and destination is memory
- **Arithmetic and logic (ALU) instructions**
  - **Add, Sub, Shift**, etc.
- **Branch instructions** (control flow instructions)
  - **Br Loc, Brz Loc2**,—unconditional or conditional branches

## Tbl. 2.1 Examples of Data Movement Instructions

Instruct.	Meaning	Machine
$\overleftarrow{\text{MOV } A, B}$	Move 16 bits from mem. loc. A to loc. B	VAX11
$\overleftarrow{\text{lwz } R3, A}$	Move 32 bits from mem. loc. A to reg. R3	PPC601
$\overleftarrow{\text{li } \$3, 455}$	Load the 32 bit integer 455 into reg. 3	MIPS R3000
$\overleftarrow{\text{mov } R4, \text{dout}}$	Move 16 bits from R4 to out port dout	DEC PDP11
$\overleftarrow{\text{IN } AL, \text{KBD}}$	Load a byte from in port KBD to accum.	Intel Pentium
$\overleftarrow{\text{LEA } L(A0), A2}$	Load address pointed to by A0 into A2	M68000

## Tbl. 2.2 Examples of ALU (Arithmetic and Logic Unit) Instructions

Instruction	Meaning	Machine
MULF A, B, C	multiply the 32-bit floating point values at mem loc'ns. A and B, store at C	VAX11
nabs r3, r1	Store abs value of r1 in r3	PPC601
ori \$2, \$1, 255	Store logical OR of reg \$ 1 with 255 into reg \$2	MIPS R3000
DEC R2	Decrement the 16-bit value stored in reg R2	DEC PDP11
SHL AX, 4	Shift the 16-bit value in reg AX left by 4 bits	Intel 8086

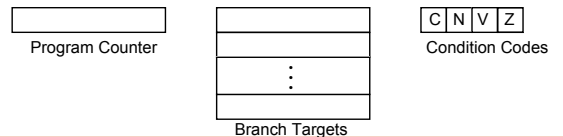
•Notice again the complete dissimilarity of both syntax and semantics

## Tbl 2.3 Examples of Branch Instructions

Instruction	Meaning	Machine
BLSS A, Tgt	Branch to address Tgt if the least significant bit of mem loc'n. A is set (i.e. = 1)	VAX11
bun r2	Branch to location in R2 if result of previous floating point computation was Not a Number (NaN)	PPC601
beq \$2, \$1, 32	Branch to location (PC + 4 + 32) if contents of \$1 and \$2 are equal	MIPS R3000
SOB R4, Loop	Decrement R4 and branch to Loop if R4 $\neq$ 0	DEC PDP11
JCXZ Addr	Jump to Addr if contents of register CX = 0.	Intel 8086

## CPU Registers Associated with Flow of Control—Branch Insts.

- Program counter usually contains the address of, or "points to" the next instruction
- Condition codes may control branch
- Branch targets may be contained in separate registers



## HLL Conditionals Implemented by Control Flow Change

- Conditions are computed by arithmetic instructions
- Program counter is changed to execute only instructions associated with true conditions

C language

```
if NUM==5 then SET=7
```

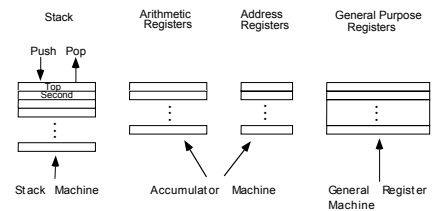
Assembly language

```

CMP.W #5, NUM ;the comparison
BNE L1 ;conditional branch
MOV.W #7, SET ;action if true
L1 ... ;action if false
    
```

## Classes of Machines

- Where the operands and result are located and how they are specified by the instruction
- CPU registers or main memory



# Hypothetical Machine Models

Fig. 2.3 The 4-Address Instruction

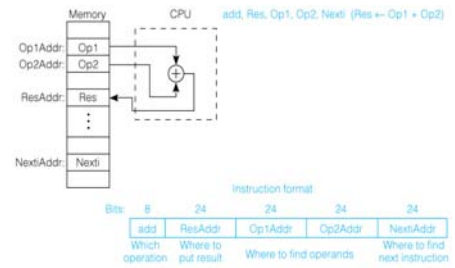


Fig 2.4 The 3 Address Instruction

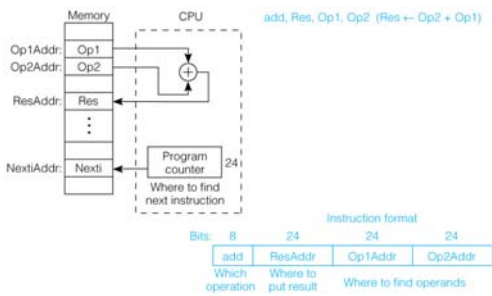


Fig. 2.5 The 2 Address Instruction

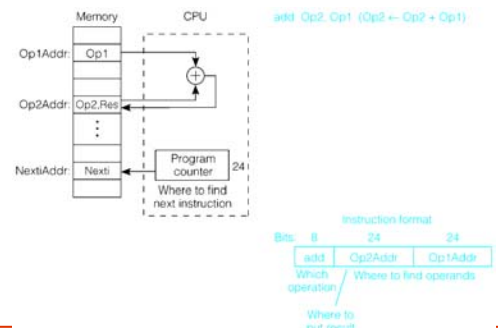


Fig. 2.6 1 Address Instructions

We now need instructions to load and store operands:  
LDA OpAddr  
STA OpAddr

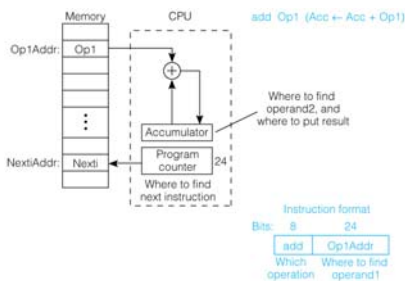
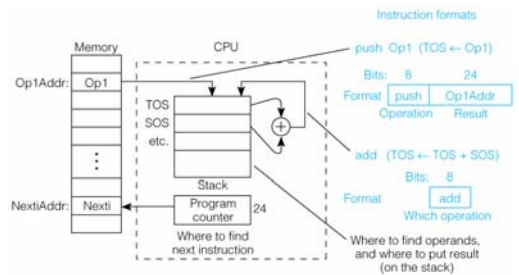


Fig. 2.7 The 0 Address Instruction



## Example 2.1 Expression evaluation for 3-0 address instructions.

Evaluate  $a = (b+c)*d-e$  for 3- 2- 1- and 0-address machines.

3-Address	2-Address	Accumulator	Stack
add a,b,c	load a,b	lda b	push b
mpy a,a,d	add a,c	add c	push c
sub a,a,e	mpy a,d	mpy d	add
	sub a,e	sub e	push d
		sta a	mpy
			push e
			sub
			pop a

## Real Machines are Not So Simple

- Most real machines have a mixture of 3, 2, 1, 0, 1 1/2 address instructions
- A distinction can be made on whether arithmetic instructions use data from memory
- If ALU instructions only use registers for operands and result, machine type is **load-store**
  - Only load and store instructions reference memory
- Other machines have a mix of register-memory and memory-memory instructions

## Tradeoffs

- Length of code sequence
- Bits per instruction
- How much information must be specified by programmer?
- Communication:
  - Registers
  - Memory

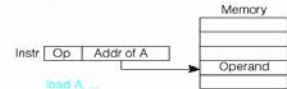
## Addressing Modes

- How many ways to access operands?

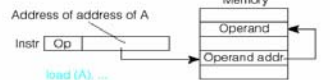
(a) **Immediate addressing:**  
instruction contains the operand



(b) **Direct addressing:**  
instruction contains address of operand



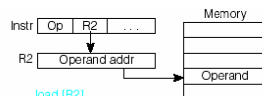
(c) **Indirect addressing:**  
instruction contains address of address of operand



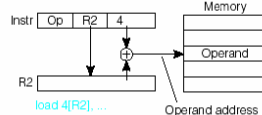
(d) **Register direct addressing:**  
register contains operand



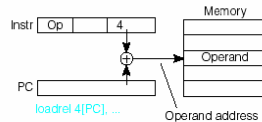
(e) **Register indirect addressing:**  
register contains address of operand



(f) **Displacement (based or indexed) addressing:**  
address of operand = register + constant

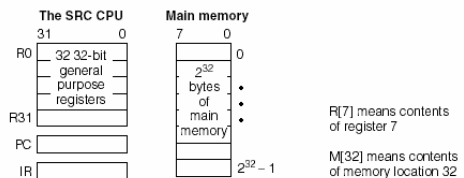


(g) **Relative addressing:**  
address of operand = PC + constant



## Example: SRC

- 32 general purpose registers of 32 bits
- 32 bit program counter, PC and instruction reg., IR
- $2^{32}$  bytes of memory address space



## SRC Basic Instruction Formats

- There are three basic instruction format types
- The number of register specifier fields and length of the constant field vary
- Other formats result from unused fields or parts

