Machine Instructions and Programs

“Must-Perform” Operations

- Data transfers between the memory and the processor registers
- Arithmetic and logic operations on data
- Program sequencing and control
- I/O transfers

Register Transfer Notation

- Identify a location by a symbolic name standing for its hardware binary address (LOC, R0,…)
- Contents of a location are denoted by placing square brackets around the name of the location ($R1 \leftarrow [LOC]$, $R3 \leftarrow [R1]+[R2]$)
- Register Transfer Notation (RTN)

Assembly Language Notation

- Represent machine instructions and programs.
- Move LOC, R1 = R1$\leftarrow$[LOC]
- Add R1, R2, R3 = R3 $\leftarrow$[R1]+[R2]

CPU Organization

- Single Accumulator
  - Result usually goes to the Accumulator
  - Accumulator has to be saved to memory quite often
- General Register
  - Registers hold operands thus reduce memory traffic
  - Register bookkeeping
- Stack
  - Operands and result are always in the stack

Instruction Formats

- Three-Address Instructions
  - ADD R1, R2, R3 $R3 \leftarrow [R1] + [R2]$
- Two-Address Instructions
  - ADD R1, R2 $R2 \leftarrow [R1] + [R2]$
- One-Address Instructions
  - ADD M $AC \leftarrow AC + [M]$
- Zero-Address Instructions
  - ADD $TOS \leftarrow [TOS] + [(TOS - 1)]$
- RISC Instructions
  - Lots of registers. Memory is restricted to Load & Store

Instruction Formats

Example: Evaluate $X = (A+B) * (C+D)$

- Three-Address
  1. ADD A, B, R1 ; $R1 \leftarrow [A] + [B]$
  2. ADD C, D, R2 ; $R2 \leftarrow [C] + [D]$
  3. MUL R1, R2, X ; $X \leftarrow [R1] * [R2]$

Instruction Formats

Example: Evaluate $X = (A+B) * (C+D)$

- Two-Address

1. MOV	A, R1	; $R1 \leftarrow [A]$
2. ADD	B, R1	; $R1 \leftarrow [R1] + [B]$
3. MOV	C, R2	; $R2 \leftarrow [C]$
4. ADD	D, R2	; $R2 \leftarrow [R2] + [D]$
5. MUL	R2, R1	; $R1 \leftarrow [R1] * [R2]$
6. MOV	R1, X	; $X \leftarrow [R1]$

Instruction Formats

Example: Evaluate $X = (A+B) * (C+D)$

- One-Address

1.	LOAD	A	; AC ← [A]
2.	ADD	B	; AC ← [AC] + [B]
3.	STORE	T	; T ← [AC]
4.	LOAD	C	; AC ← [C]
5.	ADD	D	; AC ← [AC] + [D]
6.	MUL	T	; AC ← [AC] * [T]
7.	STORE	X	; X ← [AC]

Instruction Formats

Example: Evaluate $X = (A+B) * (C+D)$

- Zero-Address
  1. PUSH A ; TOS ← [A]
  2. PUSH B ; TOS ← [B]
  3. ADD ; TOS ← [A] + [B]
  4. PUSH C ; TOS ← [C]
  5. PUSH D ; TOS ← [D]
  6. ADD ; TOS ← [C] + [D]
  7. MUL ; TOS ← (C+D)*(A+B)
  8. POP X ; X ← [TOS]

Instruction Formats

Example: Evaluate $X = (A+B) * (C+D)$

- RISC

1.	LOAD	A, R1	; R1 ← [A]
2.	LOAD	B, R2	; R2 ← [B]
3.	LOAD	C, R3	; R3 ← [C]
4.	LOAD	D, R4	; R4 ← [D]
5.	ADD	R1, R2, R1	; R1 ← [R1] + [R2]
6.	ADD	R3, R4, R3	; R3 ← [R3] + [R4]
7.	MUL	R1, R3, R1	; R1 ← [R1] * [R3]
8.	STORE	R1, X	; X ← [R1]

Using Registers

- Registers are faster
- Shorter instructions
  - The number of registers is smaller (e.g. 32 registers need 5 bits)
- Potential speedup
- Minimize the frequency with which data is moved back and forth between the memory and processor registers.

Typical Register Declaration

register int i = 10;

Note: It can not be global.