Instruction Set

Computer Organization and Architecture

SOUTH CHINA UNIVERSITY OF TECHNOLOGY
FALL 2016

DR. MAO Aihua
ahmao@scut.edu.cn

Topics

- Machine instruction characteristics
- Types of operands
- Types of operations

What is an Instruction Set?

- The complete collection of instructions that are understood and executed by a CPU
- Machine code
- Binary representation

3

Elements of an Instruction

- Operation code (Op code)
  - *What to do (Add, I/O)*
- Source Operand reference
  - *What to operate on, one or more source operands*
- Result Operand reference
  - *Where to put the answer*
- Next Instruction Reference
  - *When you have done that, where to go next*

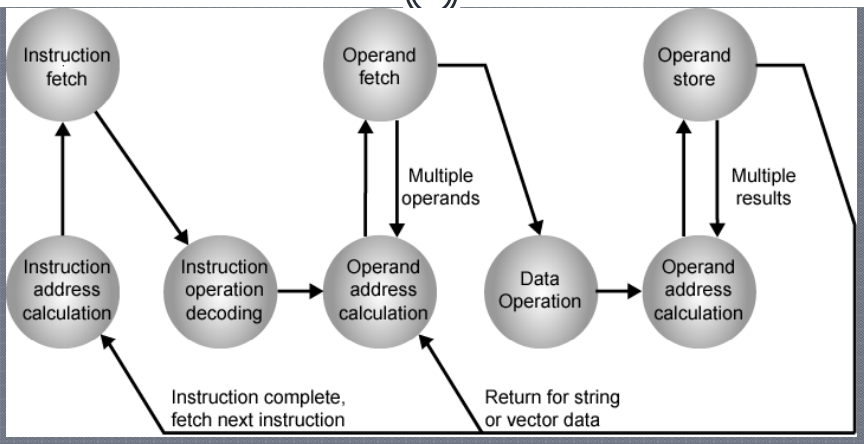
4

Where are the Operands Location?

- Think about it…
- Main memory (address must be supplied)
  - (or virtual memory or cache)
- CPU register
  - A processor contains one or more registers, referred by the machine instruction
- I/O device
  - Instruction must specify the I/O module and device for the operation

5

Instruction Cycle State Diagram



Introduced in Chapter3

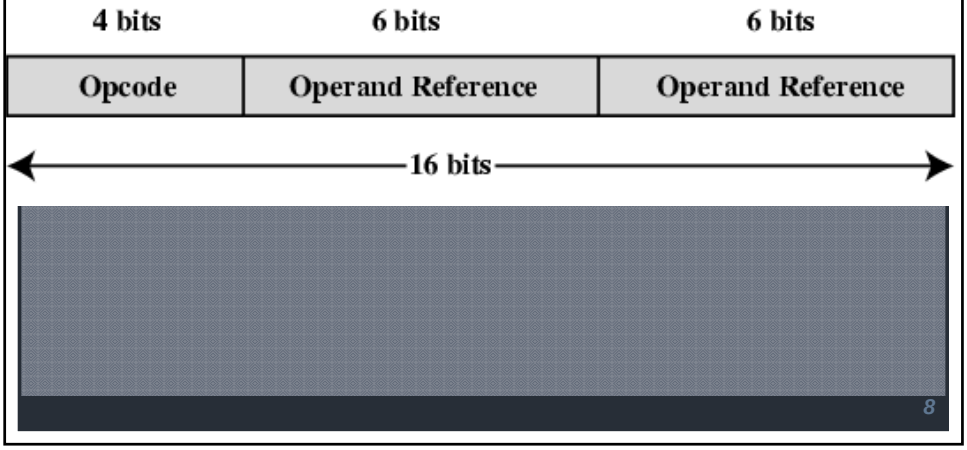
6

Instruction Representation

- In machine code, each instruction is represented by *a sequence of bits*
- The instruction is *divided into fields* (opcode, operand reference)
- During the instruction execution, an instruction is read into an *instruction register (IR)* in the processor.

7

Simple Instruction Format



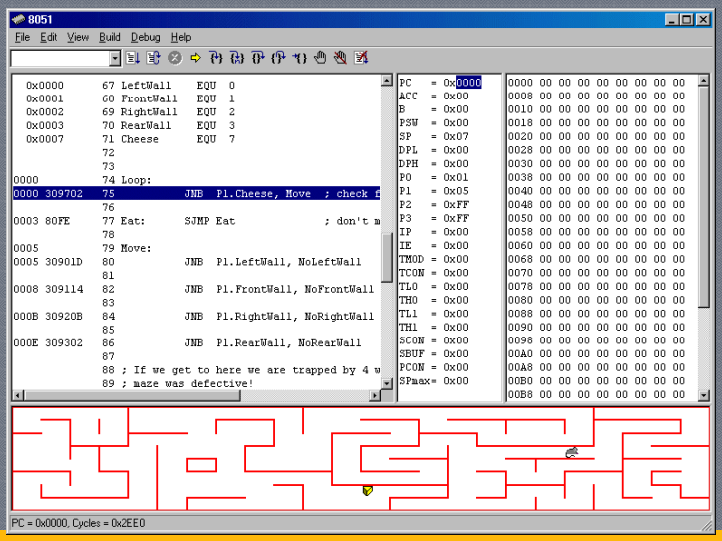
8

Instruction Representation

- It's difficult to understand the *binary machine* instructions (programmers, readers) a *symbolic representation* is used
- Opcodes are represented by abbreviations
  - e.g. *ADD, SUB, LOAD*
- Operands can also be represented symbolically
  - ADD *R,Y,* Y is the address of a location, R is a particular register

9

Program ("Assembler") accept the symbol input, convert opcodes and operands references into binary forms, construct binary machine instruction



Consider a Example

- In a high-level language instruction
- X=X+Y
- How to be accomplished with machine instruction?
- Assume X and Y correspond to locations 513 and 514
  - *Load a register with the content of 513*
  - *Add the content of 514 to the register*
  - *Store the content of register to 513*

11

How the Program is Executed?

- Any program written in high-level language must be translated into *machine language*
- The *set of machine instruction* must be *sufficient* to express any instruction from high-level language
- A machine language express operations in a basic form involving *data process and movement*

12

Instruction Types

- Data processing
  - *Arithmetic and logic instructions, deal with any type of data*
- Data storage
  - *Main memory instructions, move data between memory and registers*
- Data movement (I/O)
  - *I/O instructions, transfer data and program into memory and the results to user*
- Program flow control
  - *Test and Branch instructions*

13

Number of Addresses

- Maximum number of addresses need in an instruction?
  - *Two addresses for Operand 1, Operand 2*
  - *Third address for result*
  - *Address for next instruction (usually implicit)*
  - *However, four-address instruction is extremely rare*
- Most instructions have one, two, or three address instructions

14

Example in Different Numbers of Address

Instruction		Comment
SUB	Y, A, B	Y ← A – B
MPY	T, D, E	T ← D × E
ADD	T, T, C	T ← T + C
DIV	Y, Y, T	Y ← Y ÷ T

(a) Three-address instructions

$$Y = \frac{A - B}{C + (D \times E)}$$

Instruction		Comment
MOVE	Y, A	Y ← A
SUB	Y, B	Y ← Y – B
MOVE	T, D	T ← D
MPY	T, E	T ← T × E
ADD	T, C	T ← T + C
DIV	Y, T	Y ← Y ÷ T

(b) Two-address instructions

Instruction		Comment
LOAD	D	AC ← D
MPY	E	AC ← AC × E
ADD	C	AC ← AC + C
STOR	Y	Y ← AC
LOAD	A	AC ← A
SUB	B	AC ← AC – B
DIV	Y	AC ← AC ÷ Y
STOR	Y	Y ← AC

(c) One-address instructions

Number of Addresses

- Three addresses
  - *Addresses for Operand 1, Operand 2, Result*
  - *a=b+c*
  - *Not common because they require a relatively long instruction format to hold the three address references*

16

Number of Addresses

- Two addresses
  - *a = a + b*
  - *Reduces length of instruction*
  - *One address doubles duty as operand and result*
  - *Requires some extra work*
    - *Avoid altering the value of operand, temporary storage to hold some results*

17

Number of Addresses

- One address
  - *Implicit second address*
  - *The implicit address is usually a process register , known as accumulator (AC)*
  - *Common in early machines*

18

Number of Addresses

- 0 (zero) addresses
  - *All addresses implicit*
  - *Uses a special memory organization: stack*
  - *e.g. push a*
  - *push b*
  - *add*
  - *pop c*
  - *c = a + b*

19

Summary

Number of Addresses	Symbolic Representation	Interpretation
3	OP A, B, C	A ← B OP C
2	OP A, B	A ← A OP B
1	OP A	AC ← AC OP A
0	OP	T ← (T – 1) OP T

AC = accumulator
T = top of stack
(T – 1) = second element of stack
A, B, C = memory or register locations

How Many Addresses

- More addresses
  - *More complex (powerful?) instructions*
  - *More registers*
    - *Inter-register operations are quicker*
  - *Fewer instructions per program*
- Fewer addresses
  - *Less complex (powerful?) instructions*
  - *More instructions per program*
  - *Faster fetch/execution of instructions*

21

Instruction Set Design

- It is very complex because if affects so many aspects of the computer system
- Has a significant effect on *the functions performed* by the processor
- Is the programmers' means to control the processor

22

Instruction Set Design: design issues

- Operation repertoire
  - *How many operations?*
  - *What can they do?*
  - *How complex are they?*
- Data types:
  - *Types of data upon which operation performed*
- Instruction formats
  - *Length of op code field (in bits)*
  - *Number of addresses*

23

Instruction Set Design : design issues

- Registers
  - *Number of CPU registers available*
  - *Which operations can be performed on which registers?*
- Addressing modes (discuss later)
- RISC VS CISC

24

Types of Operand

- Addresses
- Numbers
  - *Integer/floating point*
  - *Decimal: familiar to the human users*
- Characters
  - *Can not easily stored or transmitted*
  - *ASCII code, 7-bit, 128 characters*
- Logical Data: bit data or flag
  - *Store an array of Boolean*
  - *Shift bits in floating point operations*

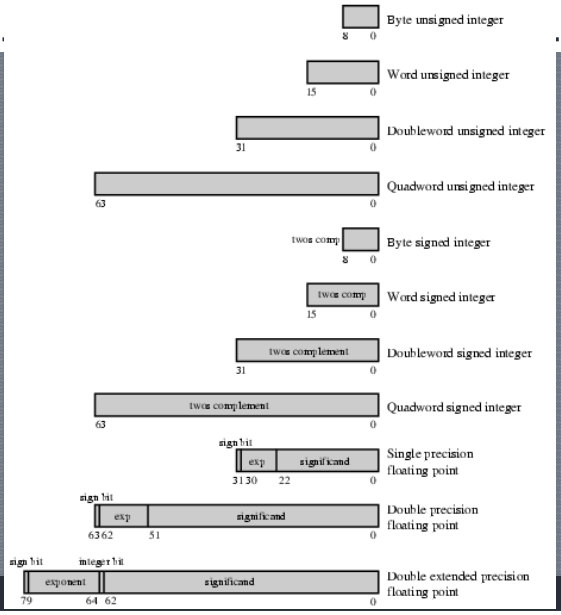
25

Pentium Data Types

- 8 bit Byte
- 16 bit word
- 32 bit double word
- 64 bit quad word
- Addressing is by 8 bit unit
- A 32 bit double word is read at addresses divisible by 4

26

Pentium Numeric Data Formats



27

Types of Operation

- Data Transfer
- Arithmetic/Logical
- Conversion
- I/O
- System Control
- Transfer of Control

28

Data Transfer

- Specify several things
  - *Source location*
  - *Destination location*
  - *Amount of data*
- May be different instructions for different movements
  - e.g. IBM S/390, listed in table 10.5
- Or one instruction and different addresses
  - e.g. VAX

29

Data Transfer

- If both source and destination are registers
  - *Simply transfer from one to another*
- If one or both operands are in memory
  - *Calculate the memory address, based on address mode*
  - *If the address refers to virtual memory, transfer from virtual to actual memory*
  - *Determine whether the addressed item is in cache*
  - *If not, issue a command to the memory module*

30

Arithmetic

- Basic arithmetic operations: *Add, Subtract, Multiply, Divide*
- Always provide for *signed Integer*
- Often also provide for *floating point*
- Others may include
  - *Increment (a++)*
  - *Decrement (a--)*
  - *Negate (-a)*
  - *Absolute*

31

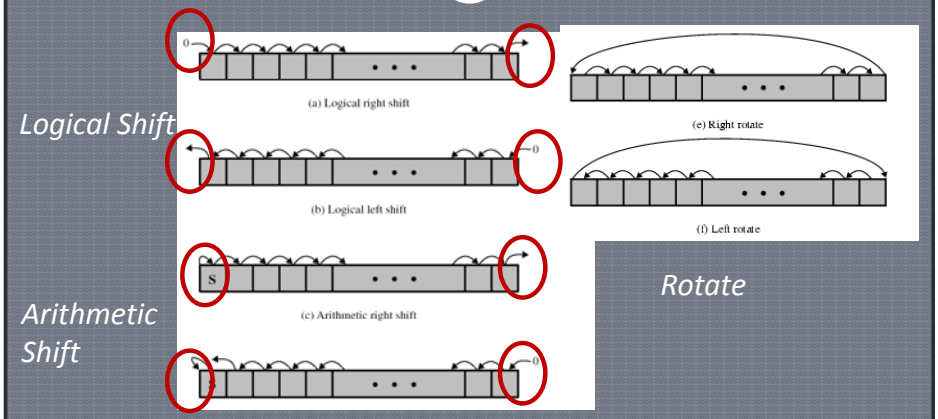
Logical

- *AND, OR, XOR, NOT*, listed in Table 10.6
- Logical operation can be applied bitwise

P	Q	NOT P	P AND Q	P OR Q	P XOR Q	P=Q
0	0	1	0	0	0	1
0	1	1	0	1	1	0
1	0	0	0	1	1	0
1	1	0	1	1	0	1

32

Shift and Rotate Operations



33

Shift and Rotate Operations

- Logical shift
  - *On one end, the bit shifted out is lost*
  - *On the other end, a 0 is shifted in*
- Arithmetic shift
  - *On a right arithmetic shift, the sign bit is replicated into the bit to this right*
  - *On a left arithmetic shift, all bits shift except the sign bit*
  - *A right arithmetic shift = a division by 2*
  - *A left arithmetic shift = a multiplication by 2*

34

Examples of Shift and Rotate operation

Table 10.7 Examples of Shift and Rotate Operations

Input	Operation	[redacted]
10100110	Logical right shift (3 bits)	
10100110	Logical left shift (3 bits)	
10100110	Arithmetic right shift (3 bits)	
10100110	Arithmetic left shift (3 bits)	
10100110	Right rotate (3 bits)	
10100110	Left rotate (3 bits)	

35

Input/Output

- May be specific instructions
- May be done using data movement instructions (*isolated programmed I/O, memory-mapped programmed I/O*)
- May be done by a separate controller (DMA)
- May be the use of an I/O processor

36

Homework

- Review questions 10.4, 10.7
- Turn in on 15th Nov.