CH10 Instruction Sets: Characteristics and Functions

- **Software and Hardware interface**
- Machine Instruction Characteristics
- Types of Operands
- Pentium II and PowerPC Data Types
- Types of Operations
- Pentium II and PowerPC Operation Types
- Assembly Language

TECH
Computer Science

What is an instruction set?

- The complete collection of instructions that are understood by a CPU
- Machine Code
- Binary
- Usually represented by assembly codes

Elements of an Instruction

- Operation code (Op code)
  - **Do this**
- Source Operand reference
  - **To this**
- Destination (Result) Operand reference
  - **Put the answer here**
- Next Instruction Reference
  - **When you have done that, do this...**

Instruction Representation

- In machine code each instruction has a unique bit pattern
- For human consumption (well, programmers anyway) a symbolic representation is used
  - **e.g. ADD, SUB, LOAD**
- Operands can also be represented in this way
  - **ADD A, B**

Instruction Types

- Data processing
- Data storage (main memory)
- Data movement (I/O)
- Program flow control

Number of Addresses (a)

- 3 addresses
  - **Operand 1, Operand 2, Result**
  - **a = b + c;**
- May be a forth - next instruction (usually implicit)
  - **Not common**
  - **Needs very long words to hold everything**

Number of Addresses (b)

- 2 addresses
  - **One address doubles as operand and result**
  - **a = a + b**
  - **Reduces length of instruction**

Number of Addresses (c)

- 1 address
  - **Implicit second address**
  - **Usually a register (accumulator)**
  - **Common on early machines**

Number of Addresses (d)

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

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**

Design Decisions (1)

- Operation repertoire
  - **How many ops?**
  - **What can they do?**
  - **How complex are they?**
- Data types
- Instruction formats
  - **Length of op code field**
  - **Number of addresses**

Design Decisions (2)

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

Types of Operand

- Addresses
- Numbers
  - **Integer/floating point**
- Characters
  - **ASCII etc.**
- Logical Data
  - **Bits or flags**
- (Aside: Is there any difference between numbers and characters? Ask a C programmer!)

Pentium Processors

- Pentium: superscaler techniques allowing multiple instructions to execute in parallel
- Pentium Pro: branch prediction, speculative execution
- Pentium II: MMX technology to process video, audio, and graphics
- Pentium III: additional floating-point instruction to support 3D graphics software.

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

Specific Data Types

- General - arbitrary binary contents
- Integer - single binary value
- Ordinal - unsigned integer
- Unpacked BCD - One digit per byte
- Packed BCD - 2 BCD digits per byte
- Near Pointer - 32 bit offset within segment
- Bit field
- Byte String
- Floating Point

Pentium Data Types

Data Formats	Range	Precision	Most significant byte … Highest addressed byte (7 0 7 0 7 0 7 0 7 0 7 0 7 0 7 0 7 0)
Word Integer	10^4	16 Bits	(Twos complement) 15 … 0
Short Integer	10^9	32 Bits	(Twos complement) 31 … 0
Long Integer	10^{18}	64 Bits	(Twos complement) 63 … 0
Packed BCD	10^{18}	18 Digits	S x $d_{17}d_{16}d_{15}d_{14}d_{13}d_{12}d_{11}d_{10}d_9d_8d_7d_6d_5d_4d_3d_2d_1d_0$ 79 … 71 … 0
Single Precision	$10^{\pm 38}$	24 Bits	S Biased Exp. Significand 31 … 22 … 0
Double Precision	$10^{\pm 308}$	53 Bits	S Biased Exponent Significand 63 … 51 … 0
Extended Precision	$10^{\pm 4932}$	64 Bits	S Biased Exponent I Significand 79 … 63 D … 0

Types of Operation

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

Data Transfer

- Specify
  - **Source**
  - **Destination**
  - **Amount of data**
- May be different instructions for different movements
  - **e.g. IBM 370**
- Or one instruction and different addresses
  - **e.g. VAX**

Arithmetic

- Add, Subtract, Multiply, Divide
- Signed Integer
- Floating point ?
- May include
  - **Increment (a++)**
  - **Decrement (a--)**
  - **Negate (-a)**

Logical

- Bitwise operations
- AND, OR, NOT

Conversion

- E.g. Binary to Decimal

Input/Output

- May be specific instructions
- May be done using data movement instructions (memory mapped)
- May be done by a separate controller (DMA)

Systems Control

- Privileged instructions
- CPU needs to be in specific state
  - **Ring 0 on 80386+**
  - **Kernel mode**
- For operating systems use

Transfer of Control //

- Branch
  - **e.g. branch to x if result is zero**
- Skip
  - **e.g. increment and skip if zero**
  - **ISZ Register1**
  - **Branch xxxx**
  - **ADD A**
- Subroutine call
  - **c.f. interrupt call**

Common Instruction Set 1

Type	*Operation Name*	*Description*
Data Transfer	Move (transfer)	Transfer word or block from source to destination
	Store	Transfer word from processor to memory
	Load (fetch)	Transfer word from memory to processor
	Exchange	Swap contents of source and destination
	Clear (reset)	Transfer word of 0s to destination
	Set	Transfer word of 1s to destination
	Push	Transfer word from source to top of stack
	Pop	Transfer word from top of stack to destination
Arithmetic	Add	Computer sum of two operands
	Subtract	Compute difference of two operands
	Multiply	Compute product of two operands
	Divide	Compute quotient of two operands
	Absolute	Replace operand by its absolute value
	Negate	Change sign of operand
	Increment	Add 1 to operand
	Decrement	Subtract 1 from operand

Common Instruction Set 2

Logical	AND	Perform the specified logical operation bitwise
	OR	
	NOT (Complement)	
	Exclusive-OR	
	Test	Test specified condition; set flag(s) based on outcome
	Compare	Make logical or arithmetic comparison of two or more operands; set flag(s) based on outcome
	Set Control Variables	Class or instructions to set controls for protection purposes, interrupt handling, timer control, etc.
	Shift	Left (right) shift operand, introducing constants at end
	Rotate	Left (right) shift operand, with wraparound end
	Jump (branch)	Unconditional transfer; load PC with specified address
	Jump Conditional	Test specified condition; either load PC with specified address or do nothing, based on condition

Common Instruction Set 3

	Jump (branch)	Unconditional transfer; load PC with specified address
	Jump Conditional	Test specified condition; either load PC with specified address or do nothing, based on condition
Transfer of Control	Jump to Subroutine	Place current program control information in known location; jump to specified address
	Return	Replace contents of PC and other register from known location
	Execute	Fetch operand from specified location and execute as instruction; do not modify PC
	Skip	Increment PC to skip next instruction
	Skip Conditional	Test specified condition; either skip or do nothing based on condition
	Halt	Stop program execution
	Wait (hold)	Stop program execution; test specified condition repeatedly; resume execution when condition is satisfied
	No operation	No operation is performed, but program execution is continued

Common Instruction Set 4

Input/Output	Input (read)	Transfer data from specified I/O port or device to destination, e.g., main memory or processor register
	Output (write)	Transfer data from specified source to I/O port or device
	Start I/O	Transfer instructions to I/O processor to initiate I/O operation
	Test I/O	Transfer status information from I/O system to specified destination
Conversion	Translate	Translate values in a section of memory based on a table of correspondences
	Convert	Convert the contents of a word from one form to another (e.g., packed decimal to binary)

Byte Order

- What order do we read numbers that occupy more than one byte
- e.g. (numbers in hex to make it easy to read)
- 12345678 can be stored in 4x8bit locations as follows

Byte Order (example)

Address	Value (1)	Value(2)
184	12	78
185	34	56
186	56	34
186	78	12

- i.e. read top down or bottom up?

Byte Order Names

- The problem is called Endian
- The system on the left has the least significant byte in the lowest address
- This is called big-endian
- The system on the right has the least significant byte in the highest address
- This is called little-endian

Standard…What Standard?

- Pentium (80x86), VAX are little-endian
- IBM 370, Moterola 680x0 (Mac), and most RISC are big-endian
- PowerPC supports both!
- Internet is big-endian
  - **Makes writing Internet programs on PC more awkward!**
  - **WinSock provides htoi and itoh (Host to Internet & Internet to Host) functions to do the convertion**