Instruction Sets

Section 10.2 Types of Operands (pp. 342-344)
Section 10.3 Pentium and PowerPC Data Types (pp. 344-346)
Section 10.4 Types of Operations (pp. 347-358)

We know that the processor operates on data. General categories of data are:

- Addresses
- Numbers
- Characters
- Logical Data

Table 10.2 Pentium Data Types

Data Type	Description
General	Byte, word (16 bits), doubleword (32 bits), and quadword (64 bits) locations with arbitrary binary contents.
Integer	A signed binary value contained in a byte, word, or doubleword, using twos complement representation.
Ordinal	An unsigned integer contained in a byte, word, or doubleword.
Unpacked binary coded decimal (BCD)	A representation of a BCD digit in the range 0 through 9, with one digit in each byte.
Packed BCD	Packed byte representation of two BCD digits; value in the range 0 to 99.
Near pointer	A 32-bit effective address that represents the offset within a segment. Used for all pointers in a nonsegmented memory and for references within a segment in a segmented memory.
Bit field	A contiguous sequence of bits in which the position of each bit is considered as an independent unit. A bit string can begin at any bit position of any byte and can contain up to $2^{32} - 1$ bits.
Byte string	A contiguous sequence of bytes, words, or doublewords, containing form zero to $2^{32} - 1$ bytes.
Floating point	See Figure 10.4.

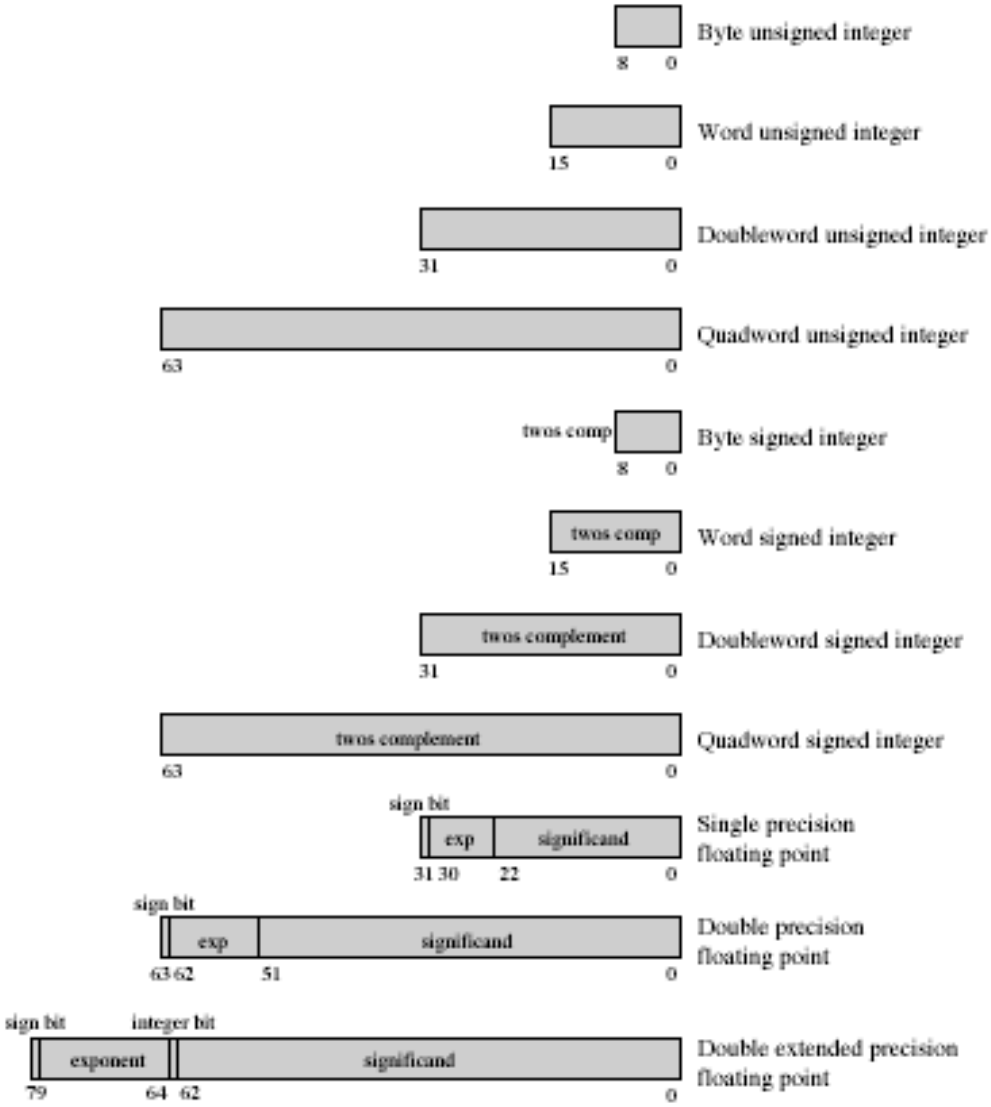

Figure 10.4 Pentium Numeric Data Formats

- The Pentium does not require that word, double-words, or quad-words be aligned on any particular boundary; however, if data is accessed across a 32-bit bus, data transfers take place in 32-bit quantities beginning with an address divisible by 4. If data is not aligned on such a boundary, then multiple transfers are needed to get the data.

- The Pentium floating-point numbers conform to the IEEE 754 standard.

- Pentium data is stored using little-endian style which means that the least significant byte is stored in the lowest address.

Q1: For the C declaration **`int intVal = -10;`** show what memory would look like if the variable **`intVal`** is located at memory location 1000. Use HEX notation.

The PowerPC is very similar to the Pentium. A few (not all) differences include:

Some instructions require that data be aligned on a 32-bit boundary

The PowerPC can use little-endian or big-endian

Q2: For the C declaration unsigned **`int intVal = 500;`** show what memory would look like if the variable **`intCal`** is located at memory location 1000. Use HEX notation.

Instruction Sets have the following common instructions:

Table 10.3 Common Instruction Set Operations (page 1 of 2)

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	Compute 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
Logical	AND	Perform logical AND
	OR	Perform logical OR
	NOT (complement)	Perform logical NOT
	Exclusive-OR	Perform logical XOR
	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 of 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

Table 10.3 Common Instruction Set Operations (page 2 of 2)

Type	Operation Name	Description
Transfer of Control	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
	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
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)