Lecture No 4

Basic Data Types

Basic Data Types

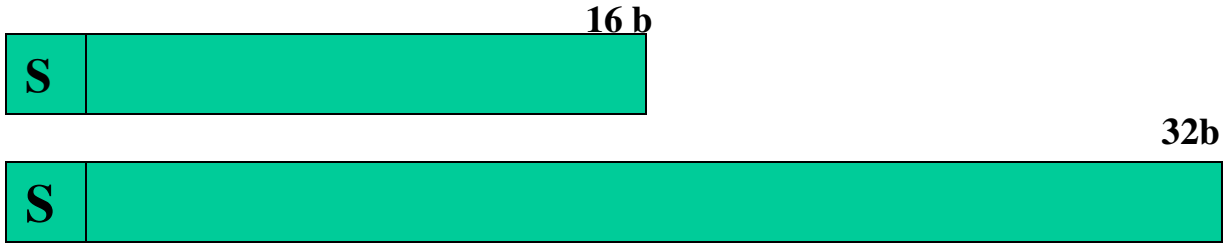
The most important aspect of an architecture is the format of data values that are operated on by the Instruction Set.

The *Data Types* defines the format and use of data objects and implies the operations that are valid for each type.

The different data types available on most machines can be broken into following classes.

1. **Integers**
2. **Floating Point (Real) Numbers**
3. **Decimal Digits**
4. **Characters**
5. **Bit / Logical**

Integers



Integers **are the fundamental data types used in computers. Different formats may be used to represent signed numbers all of which involve treating the most significant (left most) bit as sign bit. The number is treated as negative if this bit is '1'.**

•*Sign – Magnitude Representation:* This is the simplest form of representation where rightmost n-1 bits in an n bit number represent the magnitude in binary format and left most bit decides if the number is positive or negative.

+18 = 00010010
-18 = 10010010

Integers (Contd..)

Sign-Magnitude **Representation has several drawbacks like cumbersome arithmetic and two representations of Zero.**

Due to These drawbacks this is rarely used to represent integers in computers.

The most popular method of Integer Representation is called *Two's Compliment representation:* Like Sign – Magnitude representation, It also uses the most significant bit as sign bit making it easier to see if a number is positive or negative. But rest of the bits in a negative number are used as Two's compliment of the number's magnitude.

+18 = 00010010
-18 = 11101110

Reals -Fixed Point Representation

In *Fixed Point Representation* radix point is fixed (In case of Integers it is assumed to be right of right most digit.)

Same representation can be used for *Binary Fractions* by scaling the numbers so that binary point is implicitly positioned at some other location.

EXAMPLE:

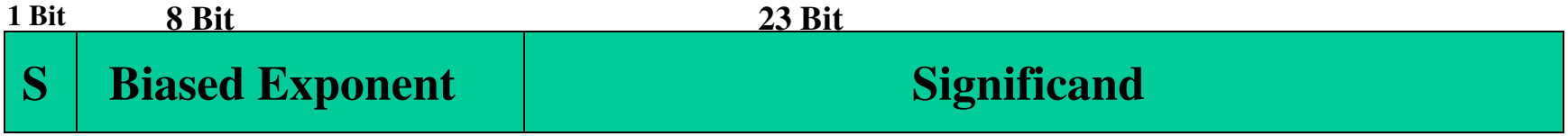
Binary Fraction 0101.01 represents

$$2^3 x0 + 2^2 x1 + 2^1 x0 + 2^0 x1 + 2^{-1} x0 + 2^{-2} x1 = 5.25$$

Reals –Floating Point Representation

For such representation *Floating Point* Format is used.

There are various binary representation of Floating Point, the most popular one has following format for a 32 bit word.



The number is stored in a binary word with following three fields.

1. **Sign : One bit field indicating positive or negative number**
2. ***Biased Exponent:* An eight bit field storing exponent plus Bias.**
3. ***Significand (Mantissa):* 23 bit field to store significand.**

Decimals

Decimal numbers are stored in two formats.

1. Packed Format: Two Digits per byte Binary Coded Decimals.

Binary Coded Decimal Representation

0	0000
1	0001
2	0010
.	
9	1001
+	1010
-	1011

Example: Number -123 0001 0010 0011 1011 in Hex #12 3b

Decimals (Contd ..)

2.Un Packed Format: One digit per byte in ASCII format.

0	0011 0000
1	0011 0001
.	
+	0010 1011
-	0010 1101
.	0010 1110

Example: -123 0011 0001 0011 0010 0011 0011 0010 1101 Hex # 31 32 33 2d

Decimals (Contd ..)

Advantages:

- **Used in calculations performed by business applications**
- **No loss of Precision by data conversion.**

Disadvantages:

- **Not Natural for most machines to perform calculations**
- **Specific instructions needed to deal with these numbers**

Characters

The character strings may be used to represent decimal or text information.

Character strings are simply a sequence of a variable number of bytes.

Bits

String of Bits (Generally limited to word size) are used to represent vectors of single bit elements, which may be tested and changed mostly using logical instructions.

The main application of bit strings is communication and control of Input / Output Devices.

Instructions

The Instruction set that defines all actions for all data types is said to have the *Orthogonal Property*.

Most machines have *Instruction sets* to perform following common core of operations.

- ***Integer Arithmetic*** **: add, subtract, multiply, divide**
- ***Floating Point arithmetic*** **: add, subtract, multiply, divide, square root**
- ***Logical*****: and, or, nor, xor, shift, rotate**
- ***Bit manipulations*****: extract, insert, test, set, clear**
- ***Control Transfer*****: jump, branch, trap**
- ***Comparison tests*****: less than or equal to, odd parity, carry**

Instructions (Contd..)

A similar format is used for branch conditions. In place of the data type specification condition code is specified.

Data Type Specifications (OP.Modifiers)

B	**Byte**	**H**	**half world**
UB	**Unsigned Byte**	**UH**	**Unsigned half word**
W	**word**	**UW**	**unsigned word**
F	**floating point**	**D**	**Double precision floating point**
C	**charcter or decimal**	**P**	**Decimal in a packed format**

Instructions (Contd..)

Branch Conditions

T	True	LE	Less than or Equal
F	False	LT	Less Than
V	Overflow	EQ	Equal
C	Carry or Borrow	NE	Not equal
PE	Even Parity	GE	Greater Than or Equal
PO	Odd Parity	GT	Greater Than

Destination Convention: ALU Instructions

Case 1: OP.X	Destination, Source 1, Source 2	(Three operand Format)
Case 2: OP.X	Destination, source	(Two Operands)
Case3: OP.X	Destination / Source 1, Source 2	(Result in Source 1 Location)