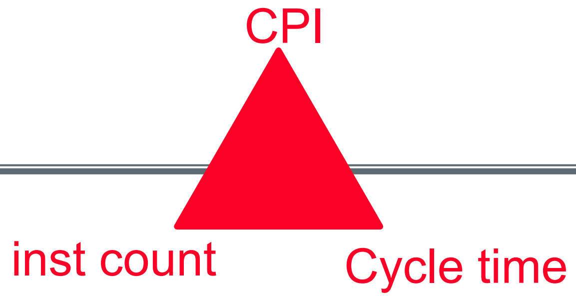
CPT_S 260 Intro to Computer Architecture
Lecture 4

Integer Representation
January 21, 2022

Ganapati Bhat
School of Electrical Engineering and Computer Science
Washington State University

Recap: Computer Performance



$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

	Inst Count	CPI	Clock Rate
Program	X		
Compiler	X	(X)	
Inst. Set.	X	X	
Organization		X	X
Technology			X

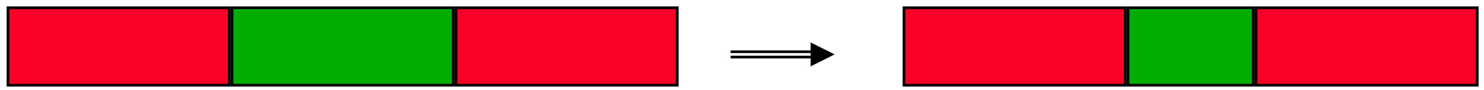
Recap: Amdahl's Law

- **How do we increase performance?**
  - Utilize parallelism
  - Principle of locality
  - Focus on the common case
- **Amdahl's law provides a method to quantify speedup**

$$Speedup_{overall} = \frac{t_{old}}{t_{new}} = \frac{1}{(1 - fraction_{enhanced}) + \frac{fraction_{enhanced}}{speedup_{enhanced}}}$$

- **Best achievable speedup is**

$$Speedup_{maximum} = \frac{1}{1 - fraction_{enhanced}}$$



Today's Topics

- Representation of numbers in different number systems
- Conversion of numbers from one system to another
- Representation of sign in binary

References

- **Computer Architecture *Chapter 3* Mano 3 edition**
  - 3-1 Data types
  - 3-2 Compliments
  - 3-3 Fixed Point Representation
- **Patterson 5E, Chapter 2**
  - 2-4 Signed and Unsigned Numbers

Introduction

- Computers store information in processor or memory registers
- This information is in binary form
- Processor register store control information – specifying a sequence of command signals needed to manipulate data in other registers
- Memory registers store binary data that are operated on (arithmetically or logically) to achieve the desired result.

Numbering Systems

- **A number system of a specific base (radix) uses numbers from 0 to that base-1**
- **Numbers can be computed to decimal through the sum of the weighted digits-**

$$\boldsymbol{Number} = \sum_{i=0}^{n} base^{i} * digit$$

	Binary	Octal	Decimal	Hexadecimal
Base	2	8	10	16
Symbols	{0,1}	{0,1,2,3,4,5,6,7}	{0,1,2,3,4,5,6,7,8,9}	{0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F}

Examples

- Decimal – $(724)_{10}$

- Binary – $(1011010100)_{2}$

- Octal – $(1324)_{8}$

- Hexadecimal – $(2D4)_{16}$

Binary

Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Binary represents numbers as a series of base-2 expressions multiplied by 0 or 1.

<u>Example:</u> 47_{10} represented in binary

$$(47)_{10}$$

$$32 + 8 + 4 + 2 + 1$$

$$1*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 1*2^1 + 1*2^0$$

$$(101111)_2$$

(Subscript notation denotes base)

Decimal → Binary

- **Choose one of two equivalent methods**
  - Subtract largest power of 2 until difference = 0. For each subtraction, place ‘1’ in a binary string at a position corresponding to the power. Fill empty positions with 0’s.
  - Divide by 2 until quotient = 0. For each division, place the remainder, either ‘0’ or ‘1’, in a binary string growing from right to left. The rightmost end of the string is the low-order bit.

Example: **Convert 131_{10} to binary**

Subtract largest power

$131 - 2^7 = 3$

$3 - 2^1 = 1$

$1 - 2^0 = 0$

$0*2^6$, $0*2^5$, $0*2^4$, $0*2^3$, $0*2^2$

Divide by 2

131 / 2 = 65 R	1
65 / 2 = 32 R	1
32 / 2 = 16 R	0
16 / 2 = 8 R	0
8 / 2 = 4 R	0
4 / 2 = 2 R	0
2 / 2 = 1 R	0
1 / 2 = 0 R	1

$(10000011)_2$

Binary → Decimal

Compose a series of base-2 terms from a binary number by identifying the position of every '1' and expressing 1's as $2^{position}$. Read binary numbers from right to left and index the rightmost position as zero.

Example: Convert 11001001_2 to decimal

$(1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1)_2$

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

$$1*2^7 + 1*2^6 + 0*2^5 + 0*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0$$

$$128 + 64 + 8 + 1$$

$$(201)_{10}$$

Range of Unsigned Binary Integers

- **Given an n-bit number**

$$x = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_1 2^1 + x_0 2^0$$

- **Range: 0 to $+2^n - 1$**
- **Example**
  - 0000 0000 0000 0000 0000 0000 0000 10112
    = 0 + … + 1×23 + 0×22 +1×21 +1×20
    = 0 + … + 8 + 0 + 2 + 1 = 1110
- **Using 32 bits**
  - 0 to +4,294,967,295

Hexadecimal

Decimal	Hex
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Hexadecimal represents numbers as a series of base-16 expressions multiplied by integers from zero to 15.

Example: **47_{10} represented in hex**

$$(47)_{10}$$

$$32 + 15$$

$$2*16^1 + 15*16^0$$

$$(2F)_{16}$$

Binary ↔ Hexadecimal

Partition binary digits right-to-left in groups of four.
Convert each group to one hexadecimal symbol.

Example: Convert 1000101010_2 to hexadecimal

$$(0010 \quad 0010 \quad 1010)_2$$

2	2	A

$$\mathbf{2*16^2 + 2*16^1 + A*16^0}$$

$$\mathbf{(22A)_{16}}$$

Hexadecimal number	Binary-coded hexadecimal
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Hexadecimal → Binary

Convert each hexadecimal symbol to a four-digit binary number and combine.

<u>Example</u>: Convert $F01_{16}$ to binary

$(F \quad 0 \quad 1)_{16}$

1111	0000	0001

$\mathbf{(111100000001)_2}$

Hexadecimal number	Binary-coded hexadecimal
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Binary ↔ Octal

Octal number	Binary-coded octal	Decimal equivalent	
0	000	0	↑
1	001	1	
2	010	2	Code
3	011	3	for one
4	100	4	octal
5	101	5	digit
6	110	6	
7	111	7	↓

Sign in Binary

- **Represent the domain of negative and positive integers in one of two distinct ways**
- **Signed Magnitude**
  - In signed-magnitude format, one bit designates sign and the remaining bits magnitude. The sign bit is the high-order bit at the leftmost position and takes ‘1’ for negative or ‘0’ for positive. The magnitude is an absolute value.
- **2’s complement**
  - In signed-2’s-complement format, positive integers take the same form as in signed-magnitude but negative integers do not. The 2’s-complement is one plus the 1’s-complement, which, for a given number and range, is the difference between the range’s maximum value and the number.

Example of the Two Methods

	Unsigned	Signed-magnitude	Signed-2's-complement
$(-15)_{10}$	N/A	N/A	N/A
$(-7)_{10}$	N/A	1111	1001
$(0)_{10}$	0000	0000	0000
$(7)_{10}$	0111	0111	0111
$(15)_{10}$	1111	N/A	N/A
Range(nbits)	$[0, 2^n - 1]$	$[-(2^{n-1} - 1), 2^{n-1} - 1]$	$[-2^{n-1}, 2^{n-1} - 1]$

2's-Complement Signed Integers

- **Given an n-bit number:**

$$x = -x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_1 2^1 + x_0 2^0$$

- **Range: -2^{n-1} to $+2^{n-1} - 1$**
- **Example**
  - $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1100_2$
    $= -1\times2^{31} + 1\times2^{30} + \ldots + 1\times2^2 + 0\times2^1 + 0\times2^0$
    $= -2{,}147{,}483{,}648 + 2{,}147{,}483{,}644 = -4_{10}$
- **Using 32 bits, 2's complement range is:**
  - –2,147,483,648 to +2,147,483,647

Signed Negation in 2's Complement

- **Complement and add 1**
  - Complement means 1 → 0, 0 → 1

- **Example: negate +2**
  - **+2 = 0000 0000 … 0010_2**
  - **–2 = 1111 1111 … 1101_2 + 1**
    = 1111 1111 … 1110_2

Signed-2's-complement

Example: (260)$_{10}$ represented in 16-bit signed-2's-complement format

$(0000\ 0001\ 0000\ 0100)_2$

Example: (-260)$_{10}$ represented in 16-bit signed-2's-complement format

$(0000\ 0001\ 0000\ 0100)_2$

$\downarrow$ Flip bits

(1	111	1110	1111	1011)$_2$
-2^{15}	2^{14} 2^{13} 2^{12}	2^{11} 2^{10} 2^{9} 2^{8}	2^{7} 2^{6} 2^{5} 2^{4}	2^{3} 2^{2} 2^{1} 2^{0})

$\downarrow$ Add one

(1	111	1110	1111	1100)$_2$
-2^{15}	2^{14} 2^{13} 2^{12}	2^{11} 2^{10} 2^{9} 2^{8}	2^{7} 2^{6} 2^{5} 2^{4}	2^{3} 2^{2} 2^{1} 2^{0})