Computer Organization and Architecture

(Lecture 6)
Computer Arithmetic- Part I

1

WHAT YOU ARE GOING TO STUDY?

- How ALU is interconnected with the rest of the processor?
- Representation of integers - sign-magnitude, ones complement, twos complement, biased
- Sign magnitude rep. - rep., drawbacks
- 2's complement rep.- characteristics, examples, geometric depiction,adv.,special cases, use of value-box for conversion, sign extension
- Arithmetic with 2's complement numbers- Addition/Subtraction/Multiplication/Division

2

ARITMETIC AND LOGIC UNIT(1)....

- Computer do not store numbers or letters
- Computers store bit sequences
- The bit sequences can be interpreted as representing integers or floating point numbers
- Arithmetic is accomplished by the direct hardware implementation of the arithmetic algorithms.

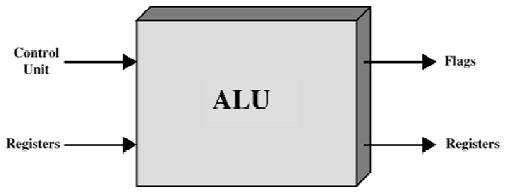
3

ARITMETIC AND LOGIC UNIT(2)

- Does the calculations
- Everything else in the computer is there to service this unit
- Handles integers
- May handle floating point (real) numbers
- May be separate Floating Point Unit (maths co-processor)

4

ALU Inputs and Outputs



5

Integer Representation

- Only have 0 & 1 to represent everything
- 8 bits word could be used to represent the non-negative numbers from 0 to 255.
- Positive numbers stored in binary
  - e.g. 41=00101001
- No minus sign and No period (radix point) for computer storage and processing
- General - n-bit sequence: $a_{n-1}\, a_{n-2} \ldots\ldots a_1 a_0$ is interpreted as unsigned integer A, then
- $A = \sum_{i=0}^{n-1} 2^i a_i$
- Representation of negative integers - Sign-Magnitude, one's complement, Two's complement, Biased

6

Sign-Magnitude

- Left most bit is sign bit
- 0 means positive
- 1 means negative
- +24 = 00011000
- -24 = 10011000
- General case A =

$$A = \sum_{i=0}^{n-2} 2^i a_i \text{ if } a_{n-1} = 0$$

$$A = -\sum_{i=0}^{n-2} 2^i a_i \text{ if } a_{n-1} = 1$$

- The rule for forming the negation of an integer is invert the sign bit.

7

Conversion Between different bit lengths

- For taking n-bit integer and store in m bits, m>n
- For sign-magnitude,move the sign bit to the new left-most position and fill in with zeros.
- +18 = 0001 0010 (sign magnitude,8bits)
- +18=0000 0000 0001 0010 (16 bits)
- -18= 1001 0010 (8 bits)
- -18=1000 0000 0001 0010 (16 bits)

8

Reasons for rarely using Sign Magnitude Rep. For implementing the integer portion of ALU-Drawbacks

- While performing addition/subtraction of numbers in sign-magnitude representation, first we have to **compare the sign of two operands and if the sign are different, then we have to compare the magnitudes of the operands** and then perform the subtraction of smaller magnitude number from the larger magnitude number and we have to put the sign of larger magnitude number as the resultant sign. --- This means the **requirement of additional hardware circuitry** needed to compare the sign as well as magnitude.Time taken for performing addition/subtraction will also be more. Example: (+60) – (-1234), (+1234) – (-60)
- **Two representations of zero** (+0 and -0)- difficult to test for zero- always should convert from -0 to +0 .

9

Twos Complement

- The two's complement representation was developed to overcome the two principal drawbacks of the sign-magnitude representation:
  - A) addition and
  - B) subtraction

10

Twos Complement

- +3 = 00000011
- +2 = 00000010
- +1 = 00000001
- +0 = 00000000
- -1 = 11111111
- -2 = 11111110
- -3 = 11111101

11

Benefits

- One representation of zero
- Arithmetic works easily (see later)
- Negating is fairly easy
  - + 3 = 00000011 (twos complement)
  - Bitwise complement = 11111100
  - Add 1 to LSB + 1
  - 11111101 = - 3

12

Twos Compliment - characteristics (1)....

❖**Range** : -2^{n-1} to $2^{n-1}-1$

8 bit 2s compliment

+127 = 01111111 = $2^7 - 1$

-128 = 10000000 = -2^7

16 bit 2s compliment

+32767 = 011111111 11111111 = $2^{15} - 1$

-32768 = 100000000 00000000 = -2^{15}

•**Number of representations of zero** : One (No negative zero).

Twos Compliment - characteristics (2a)....

⌘**Negation** (conversion from positive to negative and vice versa **)**

⌘**Process 1:** Take the Boolean complement of each bit if the integer (including the sign bit). That is, set the each 1 to 0 and each 0 to 1

⌘**Process 2:** Treating the result as an unsigned binary integer , add 1

Twos Compliment - characteristics (2b)....

The two-step process is referred to as the twos complement operation, or taking of the twos complement of an integer

Negation - example

+18	=	00010010	(two complement)
Bitwise complement	=	11101101 + 1 11101110	= -18

The negative of the negative of that number is itself : (-(-18) = 18

-18	=	11101110	(two complement)
Bitwise complement	=	00010001 + 1 00010010	= +18

Twos Compliment - characteristics (3)

⌘**Expansion of Bit length**: Add additional bit positions to the left and fill in with the value of the original sign bit (refer to slide 24)

⌘**Overflow Rule:** If two numbers with the same sign are added, then overflow occurs if and only if the result has the opposite sign. (refer to slide 26)

⌘**Subtraction Rule:** To subtract B from A, take the two's complement of B and add it to A.(refer to slide 28)

Expression for 2s complement representation

⌘ Range of positive integers represented = 0 (all the magnitude bits are 0's) through $2^{n-1}-1$ (all the magnitude bits are 1's) - Ex: for 8 bit, 00000000 (0) through 01111111(127 → 2^7-1)

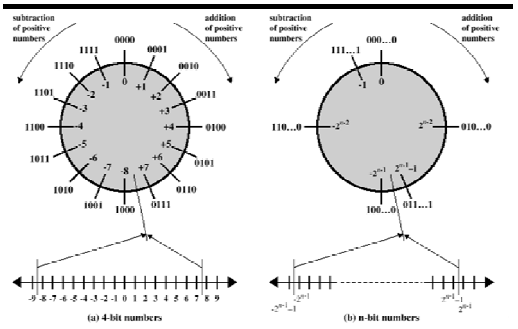
⌘ Range of negative numbers represented = -1 (11111111 (for 8bit)) through - 2^{n-1} (10000000)

⌘ Equation defining the two's complement representation for both positive and negative numbers

⌘

$$A = -2^{n-1} a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i$$

Geometric Depiction of Twos Complement Integers



Advantages-Reasons for using 2s complement rep. For the integer portion of ALU

- One representation of zero
- Arithmetic works easily - No special hardware required
- Negating is fairly easy
  - 3 = 00000011
  - Boolean complement gives 11111100
  - Add 1 to LSB 11111101
  - else
  - starting from LSB, write the bits as it is up to and including the first 1, find 1's comp. of remaining bits

19

Negation Special Case 1

- 0 = 00000000
- Bitwise complement 11111111
- Add 1 to LSB +1
- Result 1 00000000
- carry out of MSB is ignored, so:
- - 0 = 0

20

Negation Special Case 2 (1).....

- -128 = 10000000
- bitwise complement 01111111
- Add 1 to LSB +1
- Result 10000000
- So:
- -(-128) = -128 X
- Monitor MSB (sign bit)
- It should change during negation

21

Negation special case -2 - discussion(2)

- Number of different bit patterns in an n-bit word is 2^n -- even number
- wish to represent positive, negative integers and zero
- equal number of positive and negative representation of integers - two representations for zero (Sign Magnitude)
- one representation of zero, unequal number of positive and negative integers (2's comp. - rep. For - 2^{n-1} but not for + 2^{n-1}
- Range = - 2^{n-1} through 2^{n-1} -1 (example: 4 bit , -8 through +7)

22

Table 9.2 Alternative Representations for 4-bit Integers

Decimal Representation	Sign-Magnitude Representation	Two's Complement Representation	Biased Representation
+8	—	—	1111
+7	0111	0111	1110
+6	0110	0110	1101
+5	0101	0101	1100
+4	0100	0100	1011
+3	0011	0011	1010
+2	0010	0010	1001
+1	0001	0001	1000
+0	0000	0000	0111
−0	1000	—	
−1	1001	1111	0110
−2	1010	1110	0101
−3	1011	1101	0100
−4	1100	1100	0011
−5	1101	1011	0010
−6	1110	1010	0001
−7	1111	1001	0000
−8	—	1000	—

Use of a Value Box for Conversion between twos Complement Binary and Decimal

-128	64	32	16	8	4	2	1	
a) An eight position twos complement value box								
-128	64	32	16	8	4	2	1	
1	0	0	0	0	0	1	1	
-128						2	1	-125
b) Convert binary 10000011 to decimal								
128	64	32	16	8	4	2	1	
1	0	0	0	1	0	0	0	
-120				0				-120
c) Convert binary 10001000 to decimal								

24

Conversion Between different bit lengths

- For 2's complement,
- Positive number pack with leading zeros
- +18 = 00010010 (8 bits)
- +18 = 00000000 00010010 (16 bits)
- Negative numbers pack with leading ones
- -18 = 11101110 (8 bits)
- -18 = 11111111 11101110 (16 bits)
- i.e. pack with MSB (sign bit)
- called **sign extension**

25

Twos Compliment – characteristics summary

Table 9.1 Characteristics of Twos Complement Representation and Arithmetic

Range	-2^{n-1} through $2^{n-1} - 1$
Number of Representations of Zero	One
Negation	Take the Boolean complement of each bit of the corresponding positive number, then add 1 to the resulting bit pattern viewed as an unsigned integer.
Expansion of Bit Length	Add additional bit positions to the left and fill in with the value of the original sign bit.
Overflow Rule	If two numbers with the same sign (both positive or both negative) are added, then overflow occurs if and only if the result has the opposite sign.
Subtraction Rule	To subtract B from A, take the twos complement of B and add it to A.

FIXED POINT REPRESENTATION

- Called so since the binary point is fixed and assumed to be the right of the right-most digit
- can use the same representation for fractions by scaling the numbers

27

CONDITION FOR OVERFLOW

- **If two numbers are added, and they are both positive or both negative, then overflow occurs if and only if the results has the opposite sign.**
- **Note that overflow can occur whether there is a carry or not**

28

DETECTION OF OVERFLOW

- Carry out of sign bit and carry into the sign bit will be given as two inputs to the exclusive-OR gate and the output of exclusive-OR gate is connected to overflow flip-flop.
- If carry out of sign bit and carry into sign bit are equal - NO overflow
- If carry out of sign bit and carry into sign bit are not equal - OVERFLOW occurs

29

Addition and Subtraction

- Normal binary addition
- Monitor sign bit for overflow
- Take twos compliment of subtrahend and add to minuend
  - i.e. a - b = a + (-b)
- So we only need addition and complement circuits

30

Addition of Numbers in Twos Complement Representation

(-7) + (+5) 1001 +0101 1110 = -2	**(-4) + (+4)** 1100 +0100 **1** 0000 = 0
(+3) + (+4) 0011 +0100 0111 = 7	**(-4) + (-1)** 1100 +1111 **1** 1011 = -5
(+5) + (+4) 0101 +0100 1001 = overflow	**(-7) + (-6)** 1001 +1010 **1** 0011 =overflow

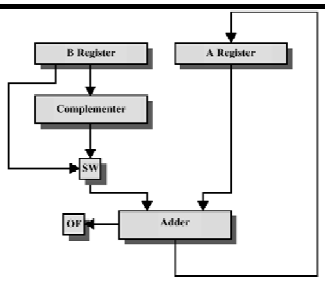
Subtraction of Numbers in Twos Complement Representation (M-S)

- To subtract one number (subtrahend) from another (minuend), take the two's complement of the subtrahend and add it to the minuend.
- Thus, subtraction is achieved using addition, as illustrated in the figure below. The last two examples demonstrate that the overflow rule still applies

Subtraction of Numbers in Twos Complement Representation (M-S) (minuend – subtrahend)

0010 +1001 1011 = -5 (negate 1011) M = 2 = 0010 S = 7 = 0111 -S = 1001	0101 +1110 1 0011 = 3 (Carry will be discarded) M = 5 = 0101 S = 2 = 0010 -S = 1110
1011 +1110 1 1001 = -7 (negate 1001) M = -5 = 1011 S = 2 = 0010 -S = 1110 (Carry will be discarded)	0101 +0010 0111 = 7 M = 5 = 0101 S = -2 = 1110 -S = 0010
0111 +0111 1110 = overflow M = 7 = 0111 S = -7 = 1001 -S = 0111	1010 +1100 1 0110 = overflow M = -6 = 1010 S = 4 = 0100 -S = 1100

Hardware for Addition and Subtraction



OF = overflow bit
SW = Switch (select addition or subtraction)

Multiplication

- Multiplication is a complex operation, whether performed in hardware or software
- We begin with the simpler problem of multiplying two unsigned (nonnegative) integers, and then we look at one of the most common techniques for multiplication of numbers in twos complement representation.
- The multiplication of two n-bit binary integers results in a product of up to 2n bit length. Example: 4 bits X 4 bits equal maximum 8 bits length

Multiplication

for unsigned binary integers
Example-paper and pencil approach

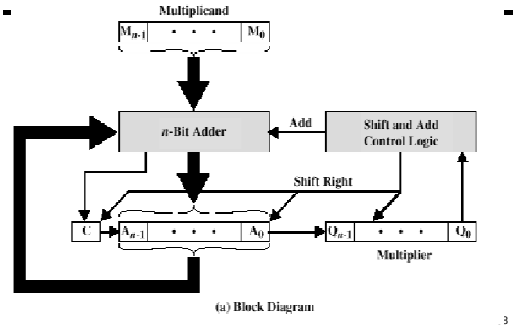
- 1011 Multiplicand (= 11 Dec.)
- x 1101 Multiplier (= 13 Dec.)
- 1011 Partial products
- 0000
- 1011
- 1011
- 10001111 Product (143 dec)
- Note: if multiplier bit is 1, copy multiplicand (place value) otherwise zero
- Note: need double length result

Multiplication-implementation

- To make the operation more efficient,
  - running addition on the partial products rather than waiting until the end - eliminates the need for storage of all partial products.
  - For each 1 on the multiplier, add and shift operation
- - for each 0, only shift - save time on the generation of partial products

37

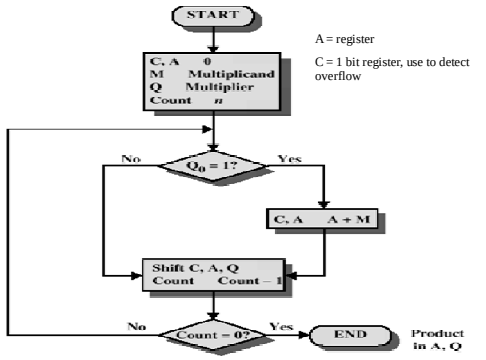
Unsigned Binary Multiplication



(a) Block Diagram

3

Flowchart for hardware implementation of Unsigned Binary Multiplication



A = register

C = 1 bit register, use to detect overflow

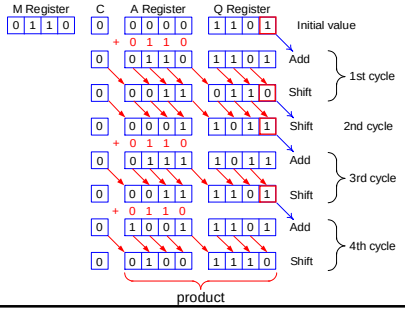
Execution of Example

C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	

40

hardware implementation of Unsigned Binary Multiplication

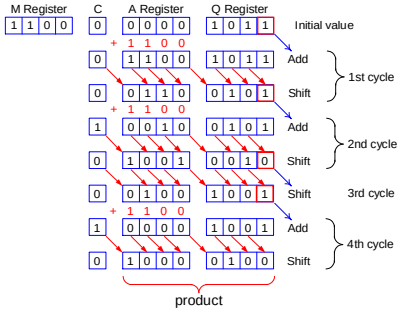
example: Consider 0110_2 x 1101_2, where 0110_2 is the multiplicand (M) and 1101_2 is the multiplier (Q)



Slides adapted from twhaw's lecture notes

41

- Consider 1100_2 x 1011_2:where 1100_2 is the multiplicand (M) and 1011_2 is the multiplier (Q)



42

PROBLEM (1)

- A. Provide a definition of one's complement numbers using a weighted sum of bits.
  B. What is the range of numbers that can be represented in one's complement?
  C. Define an algorithm for performing addition in one's complement arithmetic.

43

SOLUTION (1)

- A.
- $A = -(2^{n-1} - 1)\, a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i$
- b. from $-(2^n - 1)$ through $(2^n - 1)$
- c. 1. Add two numbers as if they were unsigned integers
  2. If there is a carry out of the sign position, then add that bit to LSB of the result and propagate the carries as necessary. (end-around carry rule)
  3. An overflow occurs if two positive numbers are added and the result is negative or if two negative numbers are added and the result is positive.

44

PROBLEM(2)

- Find the following differences using twos complement arithmetic:
- a. 111000 - 110011
- b. 11001100 - 101110
- c. 111100001111 - 110011110011
- d. 11000011 - 11101000

45

PROBLEM (3)

- Is the following a valid alternative definition of overflow in twos complement arithmetic?
  If the exclusive-or of the carry bits into and out of the left-most column is 1, then there is an overflow condition. Otherwise there is not.

46

PROBLEM (4)

- What is the range of numbers that can be represented in 8 bit and 16 bit ones complement numbers
- Perform addition and subtraction of 8 bit numbers using 1's complement representation using different examples

47

Problem (5)

- Specify the characteristics of 1's complement and sign magnitude numbers with respect to
  a. range b.number of representations of zero
  c. Negation d. Expansion of bit length
  e. Overflow Rule f. Subtraction Rule

48

Problem (6)

⌘Consider the following operation on a binary word. Start with the least significant bit, copy all bits that are 0 until the first bit is reached and copy that bit , too. Then take the complement of each bit thereafter. What is the result?

49

Problem(7)

⌘Show that the following is the equivalent definition for 2's complement.
For an n-bit integer X, the twos complement of X is formed by treating X as an unsigned integer and calculating (2^n-X).

50