EE 3613: Computer Organization

Arithmetic for Computers – 1
Number Representation & ALU

Avinash Karanth
Department of Electrical Engineering & Computer Science
Ohio University, Athens, Ohio 45701
E-mail: karanth@ohio.edu
Website: http://oucsace.cs.ohiou.edu/~avinashk/classes/ee461a/ee461a.htm
Acknowledgement: Srinivasan Ramasubramanian, Mary J. Irwin, PSU

1

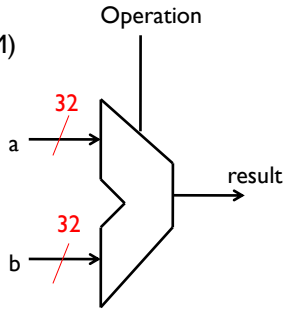
Course Administration

- Get started with Qt-SPIM ☺
- All lecture slides (including ISA) are posted
- Assignment 2 is posted – split into two parts
  - Part A is due next Friday, Sept 18
  - Part B is due the following Friday, Sept 25

2

Arithmetic

- Where we have been
  - Performance (seconds, cycles, instructions)
- Abstractions
  - Instruction Set Architecture
  - Assembly Language and Machine Language (SPIM)
- What's up ahead?
  - Implementing the architecture (Chapter 3)



3

Numbers

- Bits are just bits (no inherent meaning)
  - Conventions define relationship between bits and numbers
- Binary numbers (base 2)
  - 0000 0001 0010 0011 0100 0101 0110 0111 1000 1000
  - Decimal $0 \ldots 2^n - 1$
- Of course it gets more complicated
  - Numbers are finite (overflow)
  - Fractions and real numbers
  - Negative numbers
- How do we represent negative numbers?
  - i.e. which bit pattern will represent which numbers?

4

Possible Representations

Code	Signed Magnitude	One's Complement	Two's Complement
000	+0	+0	0
001	+1	+1	+1
010	+2	+2	+2
011	+3	+3	+3
100	-0	-3	-4
101	-1	-2	-3
110	-2	-1	-2
111	-3	-0	-1

- Issues: balance, number of zeros, ease of operation
- Which one is best? Why?

5

Number Representation

- 32 bit signed numbers
  - 0000 0000 0000 0000 0000 0000 0000 0000 = 0
  - 0000 0000 0000 0000 0000 0000 0000 0001 = +1
  - 0000 0000 0000 0000 0000 0000 0000 0010 = +2
  - ……
  - 0111 1111 1111 1111 1111 1111 1111 1110 = +2, 147,483, 646
  - 0111 1111 1111 1111 1111 1111 1111 1111 = +2, 147,483, 647
  - 1000 0000 0000 0000 0000 0000 0000 0000 = -2, 147, 483, 648
  - 1000 0000 0000 0000 0000 0000 0000 0001 = -2, 147, 483, 647
  - 1000 0000 0000 0000 0000 0000 0000 0010 = -2, 147, 483, 646
  - ……
  - 1111 1111 1111 1111 1111 1111 1111 1110 = -2
  - 1111 1111 1111 1111 1111 1111 1111 1111 = -1

6

Two's Complement Operations

- Representing positive and negative numbers
  - $(b31 \times -2^{31}) + (b30 \times 2^{30}) + (b29 \times 2^{29}) + \ldots + (b1 \times 2^{1}) + (b0 \times 2^{0})$
- Negating a two's complement number: invert all bits and add 1
  - Remember: "negate" and "invert" are quite different
- Converting n bits numbers into numbers with more than n bits
  - MIPS 16 bit immediate gets converted to 32 bits for arithmetic
  - Copy the most significant bit (sign bit) into the other bits
  - 0010 → 0000 0010
  - 1010 → 1111 1010

7

2's Complement Binary Representation

- Negate

1011

and add a 1

1010

complement all the bits

- Note: negate and invert are different!

	2'sc binary	decimal
-2^3 =	1000	-8
$-(2^3 - 1)$ =	1001	-7
	1010	-6
	1011	-5
	1100	-4
	1101	-3
	1110	-2
	1111	-1
	0000	0
	0001	1
	0010	2
	0011	3
	0100	4
	0101	5
	0110	6
$2^3 - 1$ =	0111	7

8

Addition and Subtraction

- Just like in high school (carry/borrow 1s)
- Add 6_{10} to 7_{10}

```
0 1 1 1
0 1 1 0 +
1 1 0 1
```

- Subtract 6_{10} from 7_{10}

```
0 1 1 1
0 1 1 0 -
0 0 0 1
```

- Subtract 6_{10} from 7_{10} (in two's complement)

```
0 1 1 1
1 0 1 0 +
0 0 0 1
```

- Overflow (result too large for finite computer word)
  - Eg. Adding two n-bit numbers does not yield an n-bit numbers

9

Adder: Boolean Algebra

A	B	Carry In	Carry Out	SUM
0	0	0	**0**	**0**
0	0	1	**0**	**1**
0	1	0	**0**	**1**
0	1	1	**1**	**0**
1	0	0	**0**	**1**
1	0	1	**1**	**0**
1	1	0	**1**	**0**
1	1	1	**1**	**1**

- Carry Out = A.B + B.C + A.C
- SUM = A.B.C + A'.B'.C + A'.B.C' + A.B'.C'

10

Review: Boolean Algebra and Gates

- Problem: Consider a logic function with 3 inputs: A, B and C
- Output D is true if atleast one input is true
- Output E is true if exactly two inputs are true
- Output F is true if all three inputs are true
  - Show the truth table for these three functions
  - Show the Boolean equations for these three functions
  - Show an implementation consisting of inverters, OR and AND gates

A	B	C	D	E	F
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

D =

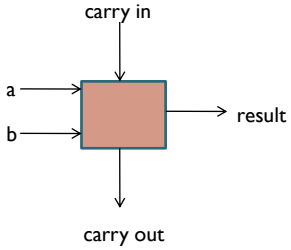
E =

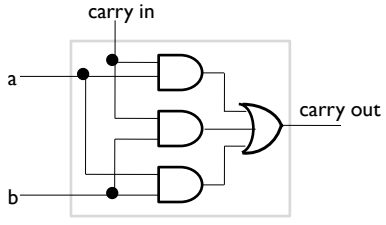
F =

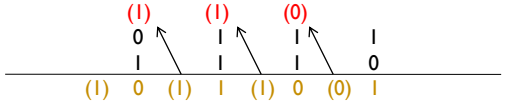
11

One-bit Adder

- Takes three input bits and generates two output bits
- Multiple bits can be cascaded



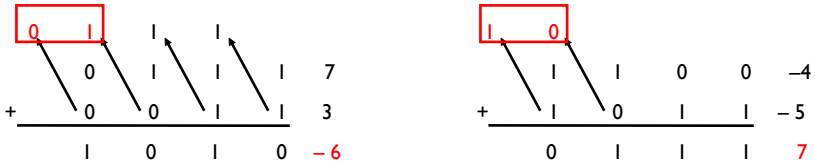




12

Detecting Overflow

- Overflow: the result is too large to represent in 32 bits
- Overflow occurs when
  - adding two positives yields a negative
  - or, adding two negatives gives a positive
  - or, subtract a negative from a positive gives a negative
  - or, subtract a positive from a negative gives a positive
- On your own: Prove you can detect overflow by:
  - Carry *into* MSB xor Carry *out of* MSB, ex for 4 bit signed numbers



13

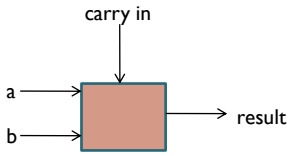
Effects of Overflow

- An exception (interrupt) occurs
  - Control jumps to predefined address for exception
  - Interrupted address is saved for possible resumption
- Details based on software system/language
- Don't always want to detect overflow
  - New MIPS instructions: addu, addiu, subu

14

An ALU (Arithmetic Logic Unit)

- Let's build an ALU to support and and or instructions
  - We will build a 1-bit ALU and use 32 of them

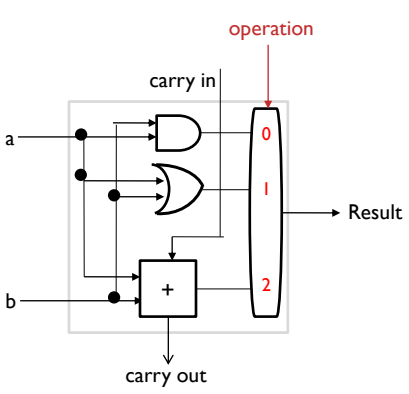


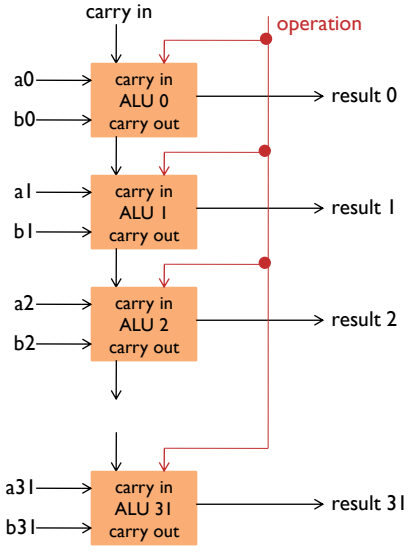
Op	A	B	Result

- Possible Implementation (sum-of-products)
  - Not easy to decide the "best" way to build something

15

Building a 32-bit ALU (AND, OR and ADD)

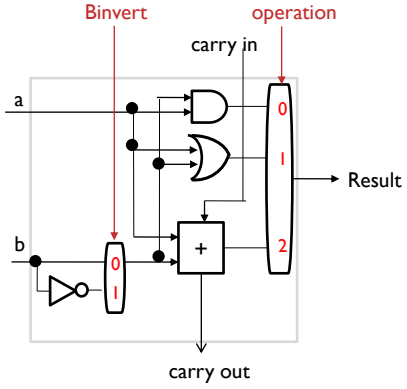


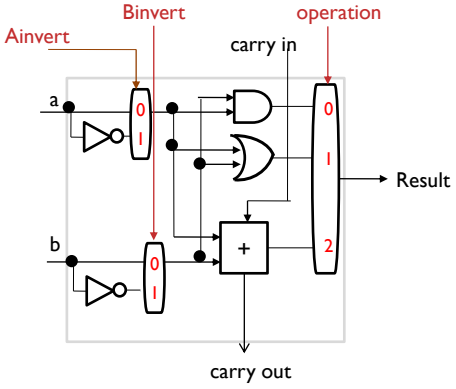


16

What about subtraction?

- Two's complement approach
  - Negate b and add
- NOR implementation
  - $\overline{(a + b)} = \bar{a} \cdot \bar{b}$

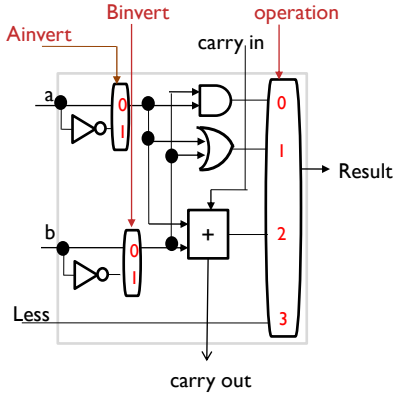


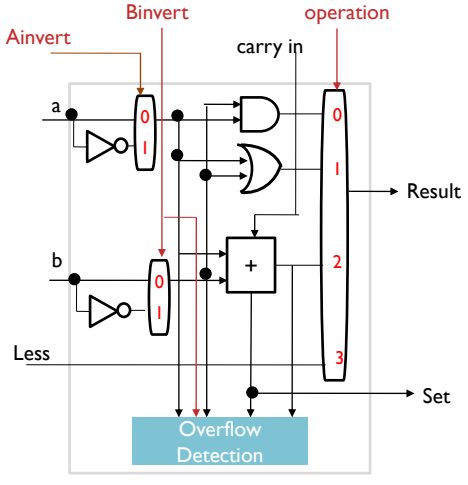


17

Supporting slt and Overflow Detection

- Can you figure out the idea?

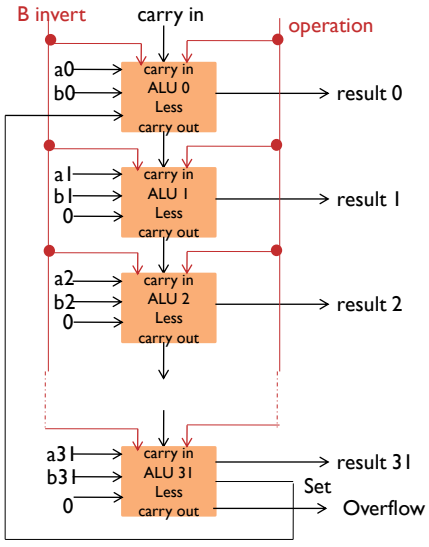




18

A 32-Bit ALU

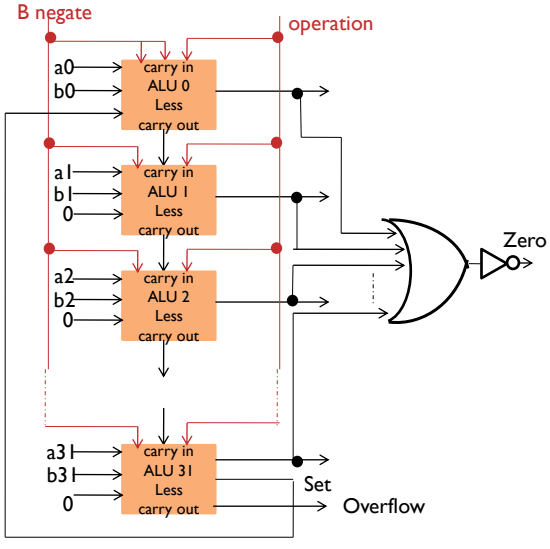
- A ripple carry ALU
- Two bits to decide
  - ADD/SUB
  - AND
  - OR
  - LESS
- A carry-in bit
  - Combine with *Binvert* to obtain *Bnegate*
- Bit 31 generates set and overflow
- How to implement branch instructions?



19

Test for Equality

- Notice the control lines
  - 000 = AND
  - 001 = OR
  - 010 = ADD
  - 110 = SUBTRACT
  - 111 = SLT



20

Conclusions

- We can build ALU to support the MIPS instruction set
  - Key Idea: Use multiplexor to select the output we want
  - Efficiently perform subtraction using two's complement
  - Replicate 1-bit ALU to produce a 32-bit ALU
- Important points about hardware
  - All of the gates are always working
  - The speed of the gate is affected by the number of inputs to the gate
  - The speed of a circuit is affected by the number of gates in series (on the critical path" or the "deepest level of logic")

21