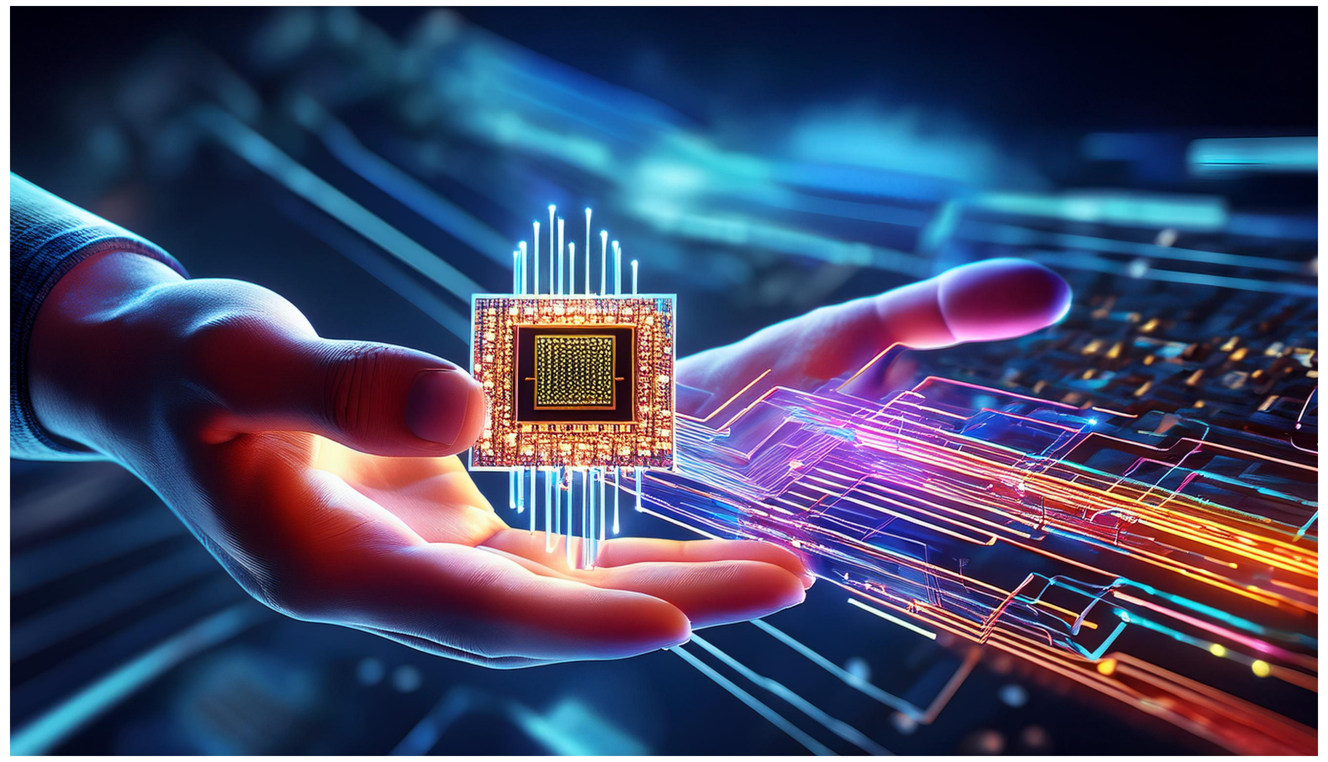
CSCI 250
Introduction to Computer Organisation
Lecture 1: Beyond Integer Arithmetics I



Jetic Gū

2024 Fall Semester (S3)

Overview

- Focus: Course Introduction
- Architecture: Logical Circuits
- Core Ideas:
  1. Review
  2. Integer Multiplication
  3. Integer Division
  4. Signed Multiplication and Division

P0
Review

From CSCI150

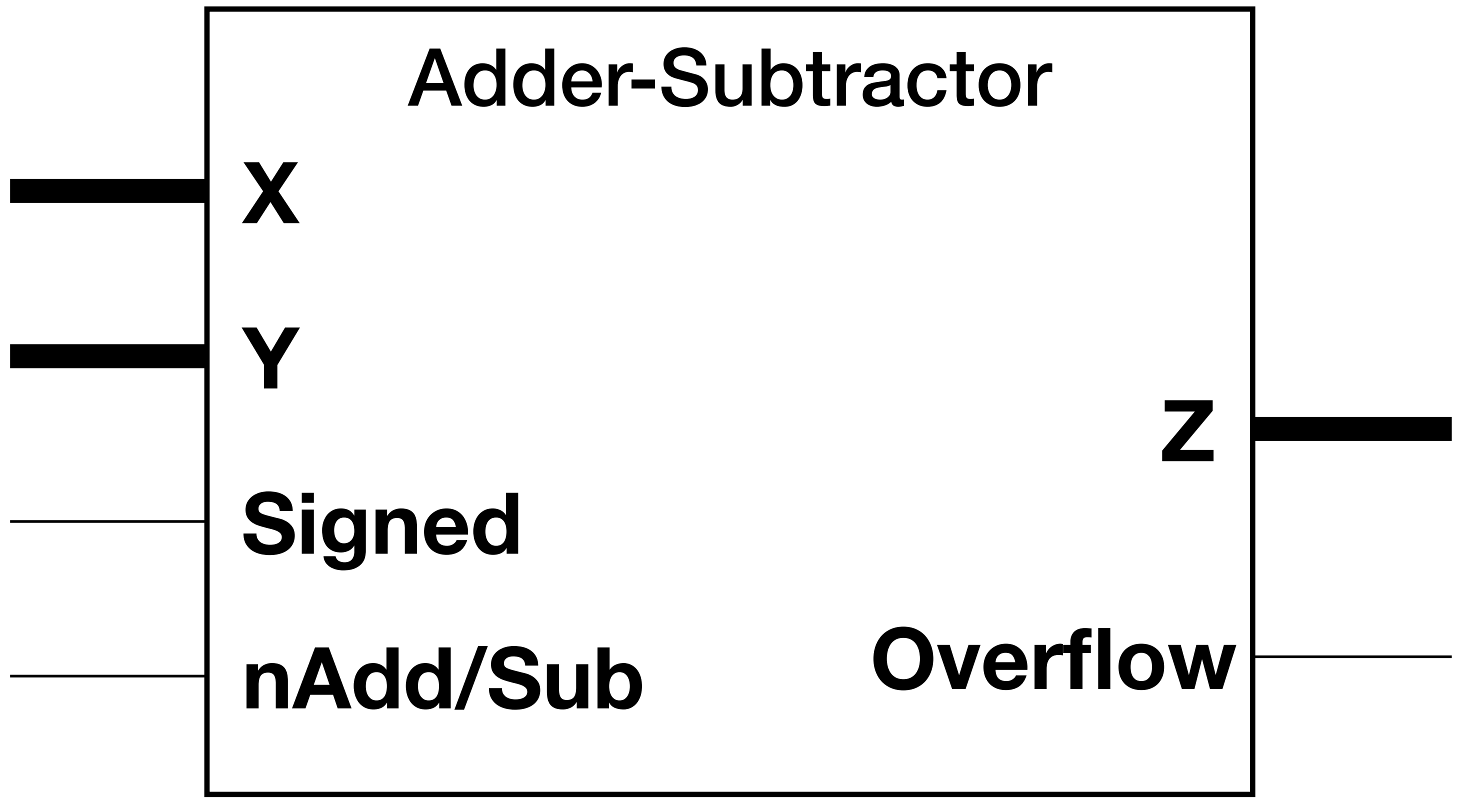
- Number systems: Binary, Hexadecimal
- Unsigned
  - Addition using adder
  - Subtraction using subtractor
  - Subtraction using adder and unsigned 2s complement
- Signed
  - Signed 2s complement

Review

P0
Review

From CSCI150

- Binary Integer Addition and Subtraction
  - A combinational arithmetic block can be designed in a hierarchical fashion
  - There's no timing element (not a sequential circuit)

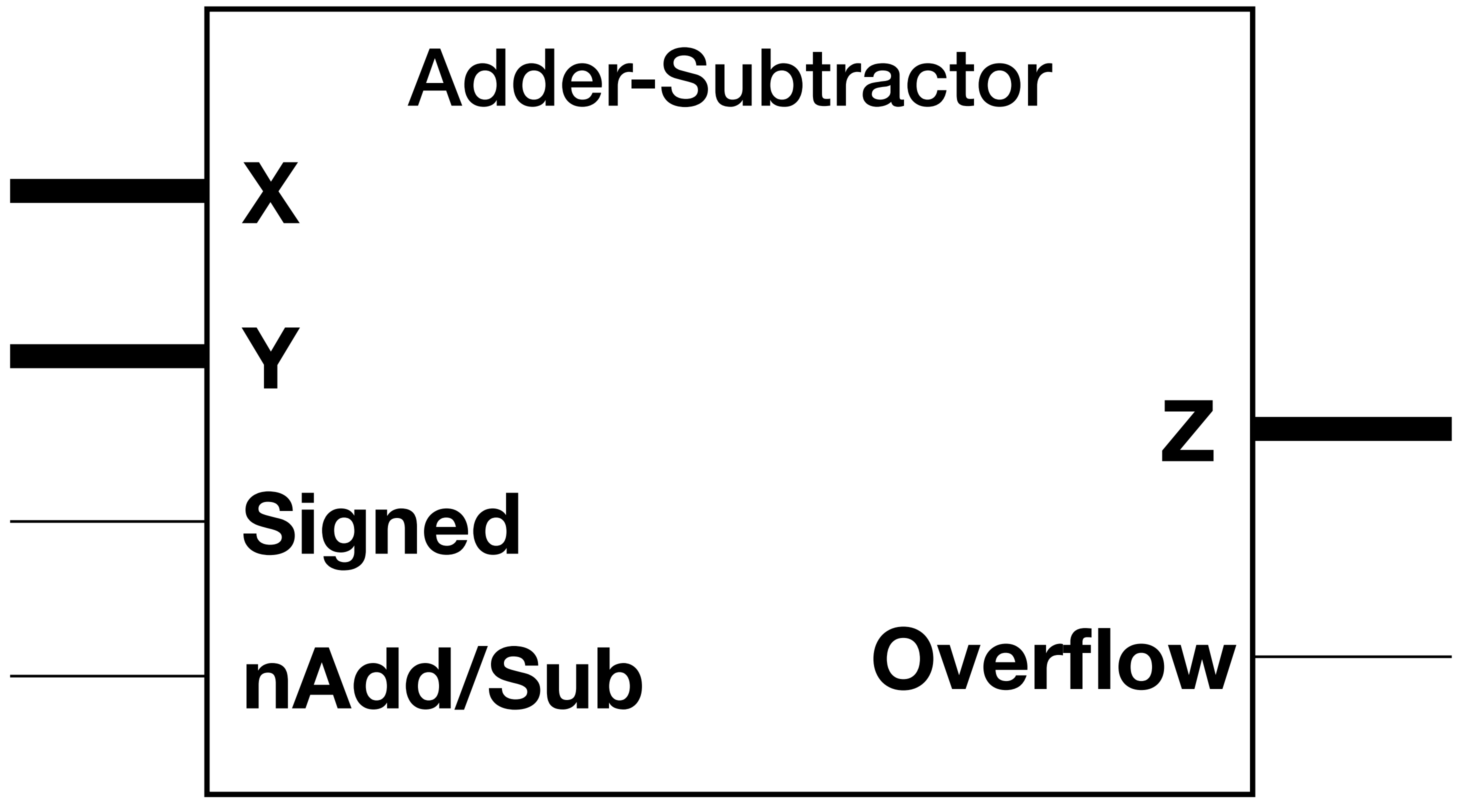


Review

P0
Review

From CSCI150

- Building blocks
  - unsigned 2s complementer
  - unsigned adder
  - unsigned subtractor*
  - multiplexer
  - XOR array
  - signed 2s complementer
  - etc…



Review

P0
Review

Problem!

- Adders and Subtractors, are relatively simple operations to design
- Multiplications and Divisions are not
  - Might require sequential circuits

Review

Integer Multiplication

P1
Multiplication

Integer Multiplication

- First, let's start with the decimal system
- Two 4 digit numbers, $X_{3:0}$ and $Y_{3:0}$, like 1 2 3 4 and 5 6 7 8

Review

P1
Multiplication

Integer Multiplication (Decimal)

$X_{3:0}$	1 2 3 4
$Y_{3:0}$	× 5 6 7 8
$X_{3:0} \times Y_0$	9 8 7 2
$X_{3:0} \times Y_1 \times 10$	8 6 3 8 0
$X_{3:0} \times Y_2 \times 10^2$	7 4 0 4 0 0
$X_{3:0} \times Y_3 \times 10^3$	+ 6 1 7 0 0 0 0
	7 0 0 6 6 5 2

$$1234 \times 8 = 9872$$
$$1234 \times 7 = 8638$$
$$1234 \times 6 = 7404$$
$$1234 \times 5 = 6170$$

Review

Integer Multiplication (Binary)

P1 Multiplication

$X_{3:0}$	1 0 1 0	
$Y_{3:0}$	× 0 1 1 0	
$X_{3:0} \times Y_0$	0 0 0 0	$1010 \times 0 = 0000$
$X_{3:0} \times Y_1 \times 2$	1 0 1 0 0	$1010 \times 1 = 1010$
$X_{3:0} \times Y_2 \times 2^2$	1 0 1 0 0 0	$1010 \times 1 = 1010$
$X_{3:0} \times Y_3 \times 2^3$	+ 0 0 0 0 0 0 0	$1010 \times 0 = 0000$
	0 1 1 1 1 0 0	

Demo

P1
Multiplication

Integer Multiplication (Binary)

$X_{3:0}$	1 0 1 0	
$Y_{3:0}$	$\times$ 0 1 1 0	
$X_{3:0} \times Y_0$	0 0 0 0	$1010 \times 0 = 0000$
$X_{3:0} \times Y_1 \times 2$	1 0 1 0 0	$1010 \times 1 = 1010$
$X_{3:0} \times Y_2 \times 2^2$	1 0 1 0 0 0	$1010 \times 1 = 1010$
$X_{3:0} \times Y_3 \times 2^3$	+ 0 0 0 0 0 0 0	$1010 \times 0 = 0000$
	0 1 1 1 1 0 0	

Demo

Integer Multiplication

- Binary integer multiplication mathematically work the same as decimal
- Binary integer multiplication is essentially $n - 1$ additions with shifters
- In addition, computers' arithmetic blocks have fixed number of bits (n)

Concept

P1
Multiplication

Integer Multiplication (4bit Binary)

$X_{3:0}$
$Y_{3:0}$

$X_{3:0} \times Y_0$
$(X_{3:0} << 1) \times Y_1$
$(X_{3:0} << 2) \times Y_2$
$(X_{3:0} << 3) \times Y_3$

4bit

```
     1 0 1 0
   × 0 1 1 0
   ---------
     0 0 0 0
   1 0 1 0 0
 1 0 1 0 0 0
+0 0 0 0 0 0 0
   ---------
 0 1 1 1 1 0 0
```

What you need to implement

$X_{3:0} \times Y_0$
+
$(X_{2:0} << 1) \times Y_1$
+
$(X_{3:0} << 2) \times Y_2$
+
$(X_{3:0} << 3) \times Y_3$

That's 3 additions for a full 4bit multiplier, and like this, you need 31 additions for a full 32bit multiplier

Are we gonna pack 31 adders into your CPU?
The answer: it's complicated

Demo

P1.2 Multiplication

Integer Multiplication (n-bit Binary)

What are the pros and cons of Option 1 & 2?

- Option 1:
  - Fully parallel combinational multiplier, use multiple adders and **multiplication-by-constant** components
  - Has much much longer propagation delay than a single n-bit adder
- Option 2:
  - Multi-step design: use storage devices, design a sequential circuit

1. Booth's multiplier, Dadda multiplier, Wallace tree, etc.

Technical

P1.2
Multiplication

Classic CPU Efficiency

	Clock cycles	1st CLK pulse	2nd+ CLK pulse
8086 Additions	2	Instruction interpretation	Perform addition
8086 Subtraction	2	Instruction interpretation	Perform subtraction
8086 Multiplication	48-60	Instruction interpretation	Goes into loop
M68000 Multiplication (32bit)	70	Instruction interpretation	Goes into loop

- 8086 (16bit) and M68000 use the multi-stage approach, where it uses a single adder and shifter to simulate the effect of having an actual Multiplier

1. http://www.righto.com/2023/03/8086-multiplication-microcode.html
2. M68000 8-/16-/32-bit Microprocessor User's Manual

Technical

P1.2
Multiplication

Modern CPU Efficiency

	Clock cycles	Result Latency
Addition / Subtraction for most modern CPUs	1-2	0 cycle
ARM CPUs (as spec)	2	4 cycles
x86-64 (2016-202X)	2	1-20 cycles*

- Modern CPUs often use paralleled approaches, which means the latency is caused by **conservative** estimation of propagation delay
- Modern CPUs tend to use RISC instructions which require little time for instruction interpretation, compared to CISC (e.g. x86, x86-64)

1. https://developer.arm.com/documentation/ddi0388/h/Cycle-Timings-and-Interlock-Behavior/Multiplication-instructions
2. x86-64 implementation differs between generations and versions of Intel/AMD designs, it's much harder to find a reliable source

Technical

Integer Division

P2
Division

Integer Division (Binary)

- A simple method of binary integer multiplication works by way of repeated shifting and adding
- A simple method of binary integer division works by doing the exact opposite: repeated shifting and subtracting

Concept

P2
Division

Integer Division (4bit Binary)

4bit

```
        0 1 1 0
×       1 0 1 0
---------------
        0 0 0 0
      0 1 1 0 0
    0 0 0 0 0 0
+ 0 1 1 0 0 0 0
---------------
  0 1 1 1 1 0 0
```

$X_{6:0}$ 0 1 1 1 1 0 0

$R_{7:0}$ 0 0 0 0 1 1 1 1 0 0

$Y_{3:0}$ 0 1 1 0

$Q_{7:0}$

Compare values by subtracting
- **if Y is greater, do not subtract and output 0**
- **if Y is lesser, subtract, output 1 and the difference goes back to R**

Demo

P2
Division

Integer Division (4bit Binary)

4bit

```
      0 1 1 0
  ×   1 0 1 0
  -----------
      0 0 0 0
    0 1 1 0 0
  0 0 0 0 0 0
+ 0 1 1 0 0 0 0
  -------------
  0 1 1 1 1 0 0
```

$X_{6:0}$ 0 1 1 1 1 0 0

$R_{9:0}$ 0 0 0 0 0 0 1 1 0 0

$Y_{3:0}$ 0 1 1 0

$Q_{7:0}$ 0

Compare values by subtracting
- **if Y is greater, do not subtract and output 0**
- **if Y is lesser, subtract, output 1 and the difference goes back to R**

Demo

P2
Division

Integer Division
(4bit Binary)

4bit

					0	1	1	0
×					1	0	1	0
					0	0	0	0
				0	1	1	0	0
			0	0	0	0	0	0
+	0	1	1	0	0	0	0	
	0	1	1	1	1	0	0	

$X_{6:0}$	0 1 1 1 1 0 0
$R_{9:0}$	0 0 0 0 0 0 0 0 0 0
$Y_{3:0}$	0 1 1 0
$Q_{7:0}$	0 0 0 1

Compare values by subtracting

- **if Y is greater, do not subtract and output 0**
- **if Y is lesser, subtract, output 1 and the difference goes back to R**

Demo

P2
Division

Integer Division (4bit Binary)

$X_{6:0}$ 0 1 1 1 1 0 0

$R_{9:0}$ 0 0 0 0 0 0 0 0 0 0

$Y_{3:0}$ 0 1 1 0

$Q_{7:0}$ 0 0 0 1 0 1 0

Compare values by subtracting

- **if Y is greater, do not subtract and output 0**
- **if Y is lesser, subtract, output 1 and the difference goes back to R**

What you need to implement

$R_{9:6} <= R_{9:6} - Y_{3:0}$ if no underflow
$R_{8:5} <= R_{8:5} - Y_{3:0}$ if no underflow
$R_{7:4} <= R_{7:4} - Y_{3:0}$ if no underflow
$R_{6:3} <= R_{6:3} - Y_{3:0}$ if no underflow
… …
$R_{3:0} <= R_{3:0} - Y_{3:0}$ if no underflow

For a 7x4bit division like this, you need 7 subtractions
For full 32bit division, like this, you need 32 subtractions

Just like multiplication, the design here is a complicated issue

Demo

P2
Division

Integer Division (n-bit Binary)

- Option 1:
  - Fully parallel combinational divider, use multiple subtractors and **division-by-constant** components
  - Has much much longer propagation delay than a single n-bit subtractor and multipliers of the same bits!
- Option 2:
  - Multi-step design: use storage devices, design a sequential circuit

Technical

P2
Division

Fun Facts

- Some Modern CPUs like PowerPC[1] do not have hardware multiplication and division for integers, they can only perform integer addition and subtractions
    - Such CPUs uses Float operations to substitute integer multiplication and division

1. https://www.ibm.com/docs/en/aix/7.3?topic=mnemonics-extended-fixed-point-arithmetic-instructions

Technical

Signed Multiplication and Division

P3
Signed

Integer
Multiplication and Division

- Recall: Decimal integer multiplication and division
  - e.g.
    (-15) * (+20) = (-1) * (+1) * (15 * 20) = -300
    (-12) / (-3) = (-1) * (-1) * (12 / 3) = 4
  - The signs are always processed separately from the numerical values
    + + -> +
    + - -> -
    - + -> -
    - - -> +

Review

P3
Signed

Signed Binary Numbers

- Signed binary numbers always have a sign bit, and n-1 magnitude bits
  e.g.
  -16 in 8bit: **1** 0010000
  +12 in 8bit: **0** 0001100

- Signed Multiplication/Division strategies
  - Separate the sign bit and magnitude bits
  - Sign bit are processed following the mul/div rules
  - Magnitude bits are processed using **unsigned Multiplier/Divider**

Concept

P3
Signed

Signed Binary Multiplication/Division

- Signed binary numbers always have a sign bit, and n-1 magnitude bits
  e.g.
  -16 in 8bit: **1** 0010000
  +12 in 8bit: **0** 0001100

Sign bit of X	Sign bit of Y	Sign of X	Sign of Y	Sign of X/Y or XY	Sign bit of X/Y or XY
0	0	+	+	+	0
0	1	+	-	-	1
1	0	-	+	-	1
1	1	-	-	+	0

Think XOR

Concept

Signed Binary Numbers

- Signed binary numbers always have a sign bit, and n-1 magnitude bits
  e.g.
  -16 in 8bit: **1** 0010000
  +12 in 8bit: **0** 0001100

- Signed Multiplication/Division strategies
  - Separate the sign bit and magnitude bits
  - Sign bit are processed **using an XOR gate**
  - Magnitude bits are processed using **unsigned Multiplier/Divider**

Concept