COMP 122/L Lecture 1

Kyle Dewey

About Me

- I research automated testing techniques and their intersection with CS education
- This is my first semester at CSUN
- Third time teaching this content

About this Class

- See something wrong? Want something improved? Email me about it! (kyle.dewey@csun.edu)
- I generally operate based on feedback

Bad Feedback

- This guy sucks.
- This class is boring.
- This material is useless.

Good Feedback

- This guy sucks, *I can't read his writing.*
- This class is boring, *it's way too slow.*
- This material is useless, *I don't see how it relates to anything in reality.*

- I can't fix anything if I don't know what's wrong

Class Motivation

```
public static void
main(String[] args) {
   ...
}
```

```
public static void
main(String[] args) {
  ...
}
```

```
public static void
main(String[] args) {
   ...
}
```

3.14956

```
public static void
main(String[] args) {
   ...
}
```

3.14956

```
public static void
main(String[] args) {
   ...
}
```

More Efficient Algorithms

3.14956

```
public static void
main(String[] args) {
    ...
}
```

More Efficient Algorithms

3.14956

Why are things still slow?

The magic box isn’t so magic

Array Access

`arr[x]`

- Constant time! (O(1))
- Where the **random** in random access memory comes from!

Array Access

`arr[x]`

- Constant ti
- Where the dom access memory co

Array Access

- Memory is loaded as chunks into *caches*
  - Cache access is much faster (e.g., 10x)
  - Iterating through an array is fast
  - Jumping around any which way is slow
- Can make code *exponentially* faster

Instruction Ordering

```
int x = a + b;
int y = c * d;
int z = e - f;
```

```
int z = e - f;
int y = c * d;
int x = a + b;
```

Instruction Ordering

```
int x = a + b;
int y = c * d;
int z = e - f;
```

3 Milliseconds?

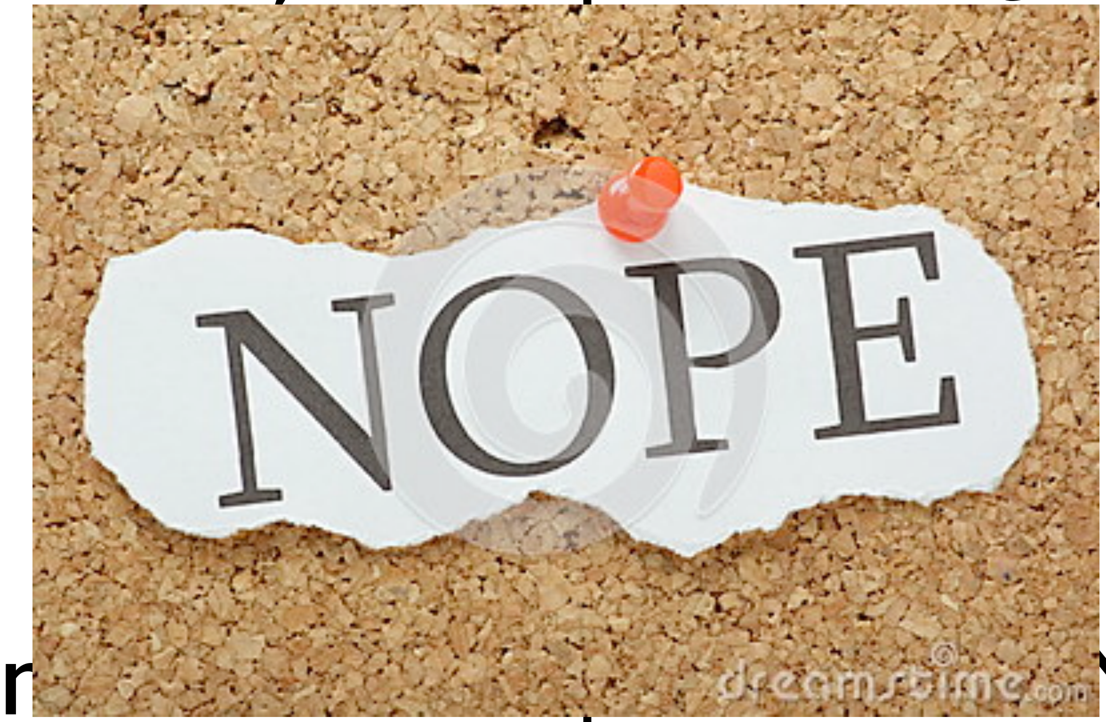
```
int z = e - f;
int y = c * d;
int x = a + b;
```

3 Milliseconds?

Instruction Ordering

```
int x = a + b;
int y = c * d;
int z = e
```

```
int z = e - f;
int y = c * d;
    x = a + b;
```



3 Millisecon

Milliseconds?

Instruction Ordering

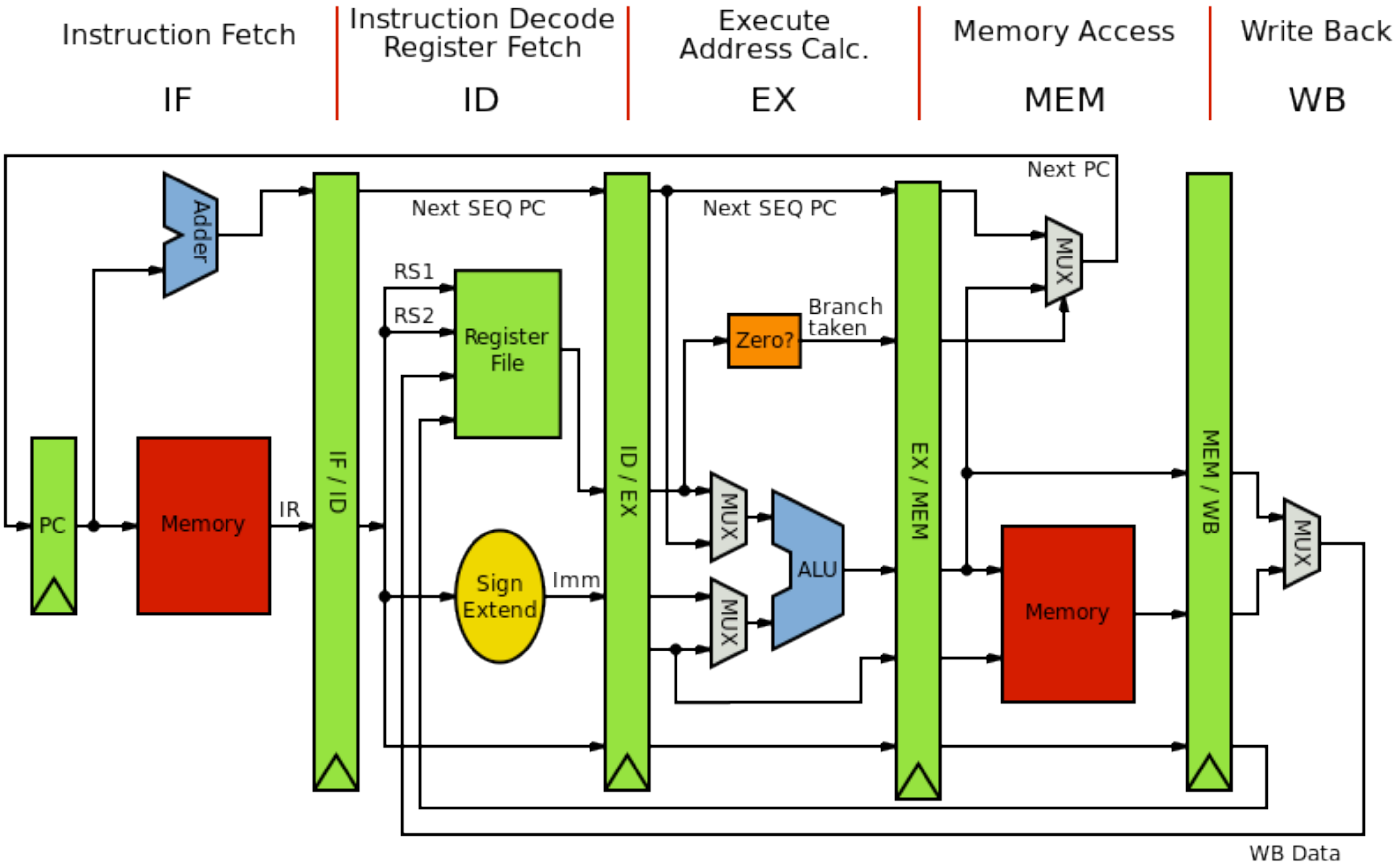
- Modern processors are *pipelined*, and can execute sub-portions of instructions in parallel
  - Depends on when instructions are encountered
- Some can execute whole instructions in different orders
- If your processor is from Intel, it is insane.

The Point

- If you really want performance, you need to know how the magic works
  - "But it scales!" - empirically, probably not
  - Chrome is fast for a reason
- If you want to write a naive compiler, you need to know some low-level details
- If you want to write a *fast* compiler, you need to know *tons* of low-level details

So Why Circuits?

So Why Circuits?



So Why Circuits?

- Basically, circuits are the programming language of hardware
  - Yes, everything goes back to physics

Overall Course Structure

Syllabus

Working with Different Bases

What's In a Number?

- Question: why exactly does 123 have the value 123? As in, what does it *mean*?

What’s In a Number?

123

What’s In a Number?

1	2	3

What's In a Number?

1	2	3
Hundreds	Tens	Ones

What's In a Number?

1	2	3
Hundreds	Tens	Ones
100	10 10	1 1 1

Question

- Why did we go to tens? Hundreds?

1	2	3
Hundreds	Tens	Ones
100	10 10	1 1 1

Answer

- Because we are in decimal (base 10)

1	2	3
Hundreds	Tens	Ones
100	10 10	1 1 1

Another View

123

Another View

1	2	3

Another View

1	2	3
1×10^2	2×10^1	3×10^0

Conversion from Some Base to Decimal

- Involves repeated division by the value of the base
  - From right to left: list the remainders
  - Continue until 0 is reached
  - Final value is result of reading remainders from bottom to top
- For example: what is 231 decimal to decimal?

Conversion from Some Base to Decimal

Conversion from Some Base to Decimal

	Remainder
$10 \,\lfloor\, \underline{231}$	
23	1

Conversion from Some Base to Decimal

	Remainder
10 | 231	
10 | 23	1
2	3

Conversion from Some Base to Decimal

	Remainder
10 ⌊231	
10 ⌊23	1
10 ⌊2	3
0	2

Now for Binary

- Binary is base 2
- Useful because circuits are either on or off, representable as two states, 0 and 1

Now for Binary

1010

Now for Binary

1	0	1	0

Now for Binary

Eights	Fours	Twos	Ones
1	0	1	0

Now for Binary

1	0	1	0
Eights	Fours	Twos	Ones
1×2^3	0×2^2	1×2^1	0×2^0
8	0	2	0

Question

- What is binary 0101 as a decimal number?

Answer

- What is binary 0101 as a decimal number?
  - 5

0	1	0	1
Eights	Fours	Twos	Ones
0×2^3	1×2^2	0×2^1	1×2^0
0	4	0	1

From Decimal to Binary

- What is decimal 57 to binary?

From Decimal to Binary

57

From Decimal to Binary

	Remainder
$2 \,\lfloor\, 57$	
28	1

From Decimal to Binary

	Remainder
$2 \,\underline{\big\rvert\, 57}$	
$2 \,\underline{\big\rvert\, 28}$	1
14	0

From Decimal to Binary

	Remainder
2 | 57	
2 | 28	1
2 | 14	0
7	0

From Decimal to Binary

	Remainder
2 | 57	
2 | 28	1
2 | 14	0
2 | 7	0
3	1

From Decimal to Binary

	Remainder
2 | 57	
2 | 28	1
2 | 14	0
2 | 7	0
2 | 3	1
1	1

From Decimal to Binary

	Remainder
2 ⌊57	
2 ⌊28	1
2 ⌊14	0
2 ⌊7	0
2 ⌊3	1
2 ⌊1	1
0	1

Hexadecimal

- Base 16
- Binary is horribly inconvenient to write out
- Easier to convert between hexadecimal (which is more convenient) and binary
  - Each hexadecimal digit maps to four binary digits
  - Can just memorize a table

Hexadecimal

- Digits 0-9, along with A (10), B (11), C (12), D (13), E (14), F (15)

Hexadecimal Example

- What is 1AF hexadecimal in decimal?

Hexadecimal Example

I	A	F

Hexadecimal Example

1	A	F
Two-fifty-sixes	Sixteens	Ones

Hexadecimal Example

1	A	F
Two-fifty-sixes	Sixteens	Ones
1×16^2	10×16^1	15×16^0

Hexadecimal Example

1	A	F
Two-fifty-sixes	Sixteens	Ones
1×16^2	10×16^1	15×16^0
	16 16 16 16 16	1 1 1 1 1
	16 16 16 16 16	1 1 1 1 1
256	(160)	1 1 1 1 1
		(15)

Hexadecimal to Binary

- Previous techniques all work, using decimal as an intermediate
- The faster way: memorize a table (which can be easily reconstructed)

Hexadecimal to Binary

Hexadecimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Hexadecimal	Binary
8	1000
9	1001
A (10)	1010
B (11)	1011
C (12)	1100
D (13)	1101
E (14)	1110
F (15)	1111