Systems and machine code

- Processors and architecture (**today**)
- Assembly (Wednesday)
- (Lab: Adapter pattern (Thursday))
- Function call and return (Friday and next week Monday)
- Function pointers (next week Wednesday)

Today:

- Model of machine execution
- The structure of assembly instructions
- First assembly examples

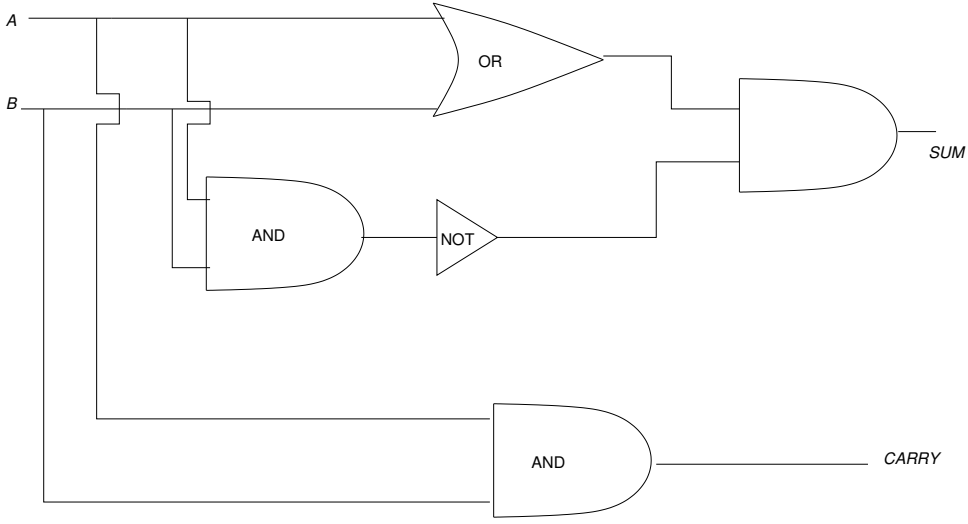
```
public class Circle {
    private int centerX, centerY;
    private double radius;
    private Color color;
    private boolean opaque;

    public Circle(int centerX, int centerY,
                  double radius, Color color,
                  boolean opaque) {
        this.centerX = centerX;
        this.centerY = centerY;
        this.radius = radius;
        this.color = color;
        this.opaque = opaque;
    }
}

Circle c = new Circle(12, -7, 8.5,
                      Color.RED, true);
```

```
struct circle
{
  int center_x, center_y;
  double radius;
  unsigned long color;
  int opaque;
};

struct circle *c = (struct circle *)
         malloc(sizeof(struct circle));
c->center_x = 12;
c->center_y = -7;
c->radius = 8.5;
c->color = 255;
c-> opaque = 1;
```


A
B
OR
AND
NOT
SUM
AND
CARRY

	Mnemonic	Operands	Description	Example
1	`MOVI`	`imm rd`	Copies the immediate value `imm` into register #`rd`.	`MOVI 5 R2` `// R2 = 5`
3	`ADD`	`rs rd`	Adds the value in register #`rs` into register #`rd`.	`ADD R1 R2` `// R2 = R2 + R1`
9	`OUT`	`rs`	Outputs the value in register #`rs` by printing it to the screen.	`OUT R1` `// PRINT R1`
10	`HALT`		Ends the program.	`HALT` `// QUIT`

	Mnemonic	Operands	Description	Example
2	`MOV`	`rs rd`	Copies the value in register #`rs` into register #`rd`.	`MOV R3 R4` `// R4 = R3`
4	`SUB`	`rs rd`	Subtracts the value in register #`rs` from register #`rd`.	`SUB R1 R2` `// R2 = R2 - R1`
5	`MUL`	`rs rd`	Multiplies register #`rd` by the value in register #`rs`.	`MULL R1 R2` `// R2 = R2 * R1`
6	`IDIV`	`rs rd`	Divides register #`rd` by the value in register #`rs` (as `int`s).	`IDIV R1 R2` `// R2 = R2 / R1`

	Mnemonic	Operands	Description	Example
7	JMP	ra	"Jump." The next instruction to be executed is found at the address in register #ra.	JMP R1 // GOTO R1
8	JNZ	rs ra	If the value of register #rs is nonzero, jumps to the address in register #ra. Otherwise, does nothing and continues normally to the next instruction.	JNZ R1 R2
17	LDLO	imm rd	Loads a commandline argument indexed by negative numbers, i.e., -1 is the last argument, -2 is the second-to-last, etc.	LDLO -1 R1

Coming up:

- **Due Wed, Apr 22.** *Do Project 7, no-ifs calculator*
- **Due Thurs, Apr 23.** *Read prelab reading and take quiz (Coming soon. . .)*

- **Due Fri, May 1.** *Do Project 8, GCD in pseudo-assembly (Also coming soon. . .)*