Computer Architecture

Nand2Tetris

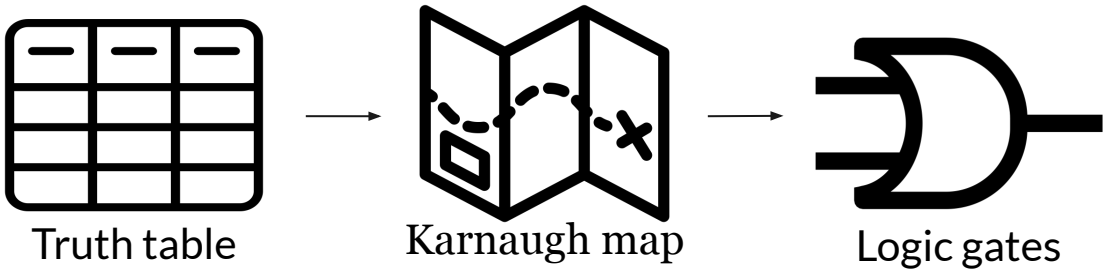
Build Tetris game from only NAND gates

Duc Nguyen Quang
Student ID: 1810118

Project 1: Boolean Logic

Basic Logic Gates:

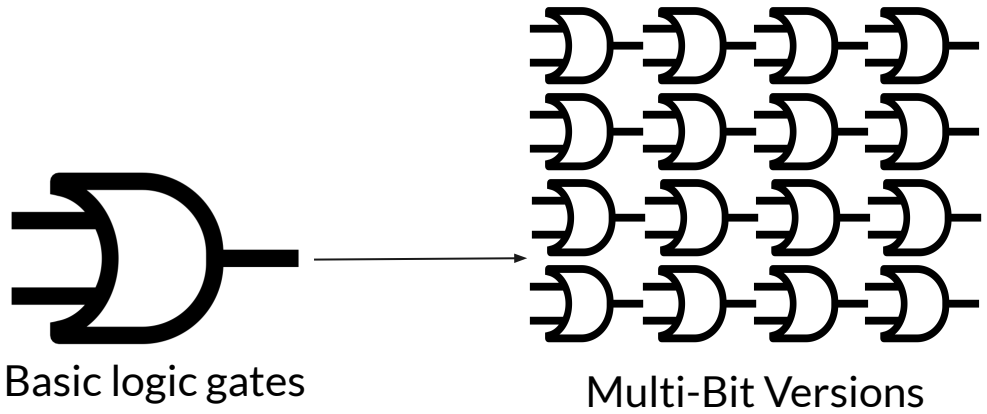
- And
- Or
- Not
- Xor
- Multiplexor
- Demultiplexor



Project 1: Boolean Logic

Multi-Bit Versions of Basic Gates:

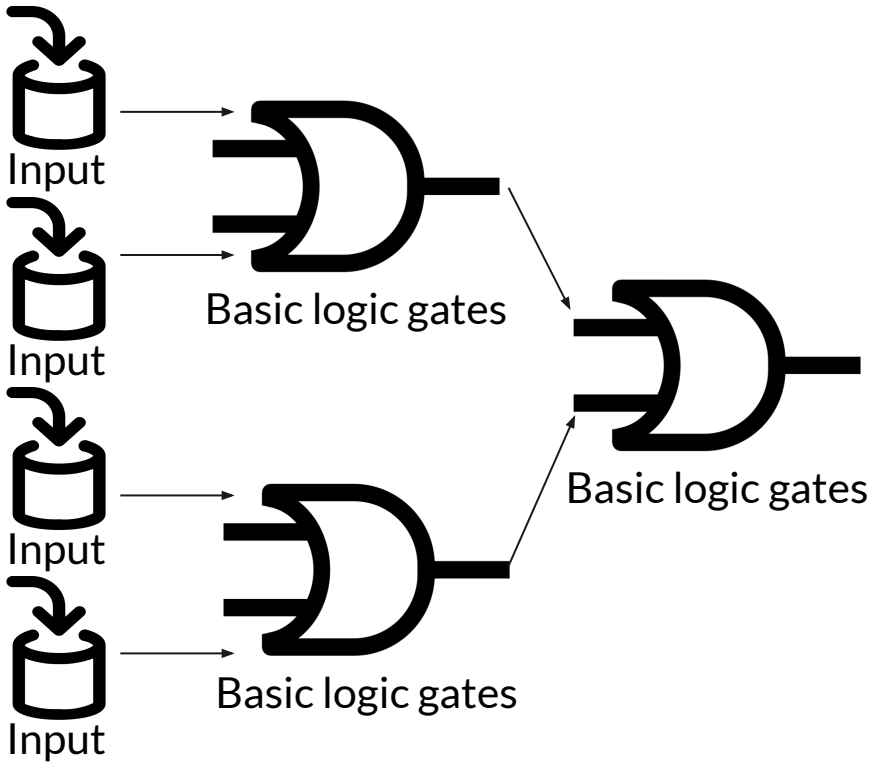
- Multi-Bit Not
- Multi-Bit And
- Multi-Bit Or
- Multi-Bit Multiplexor



Project 1: Boolean Logic

Multi-Way Versions of Basic Gates:

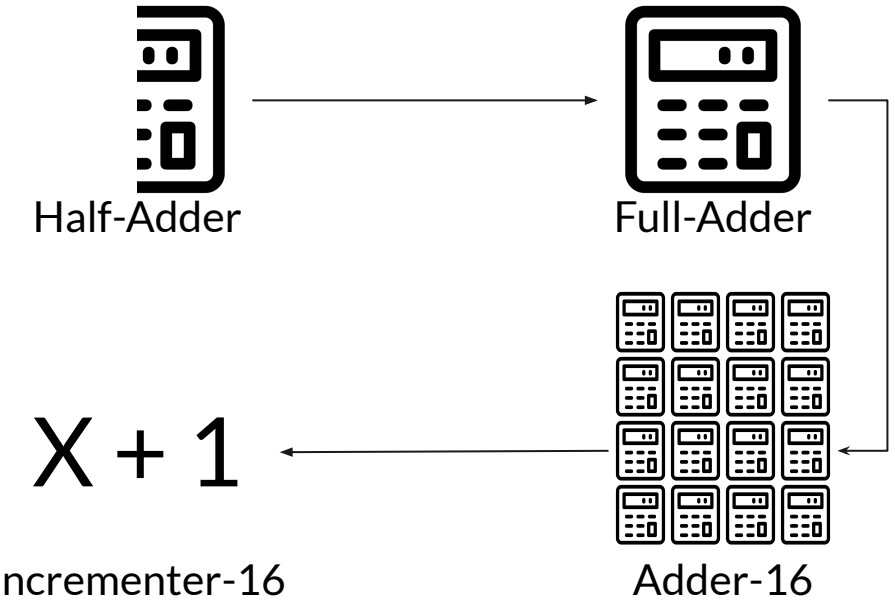
- Multi-Way Or
- Multi-Way Multiplexor
- Multi-Way Demultiplexor



Project 2: Boolean Arithmetic

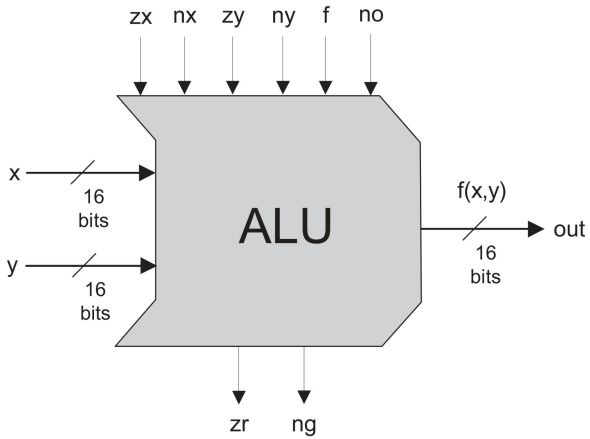
Adders:

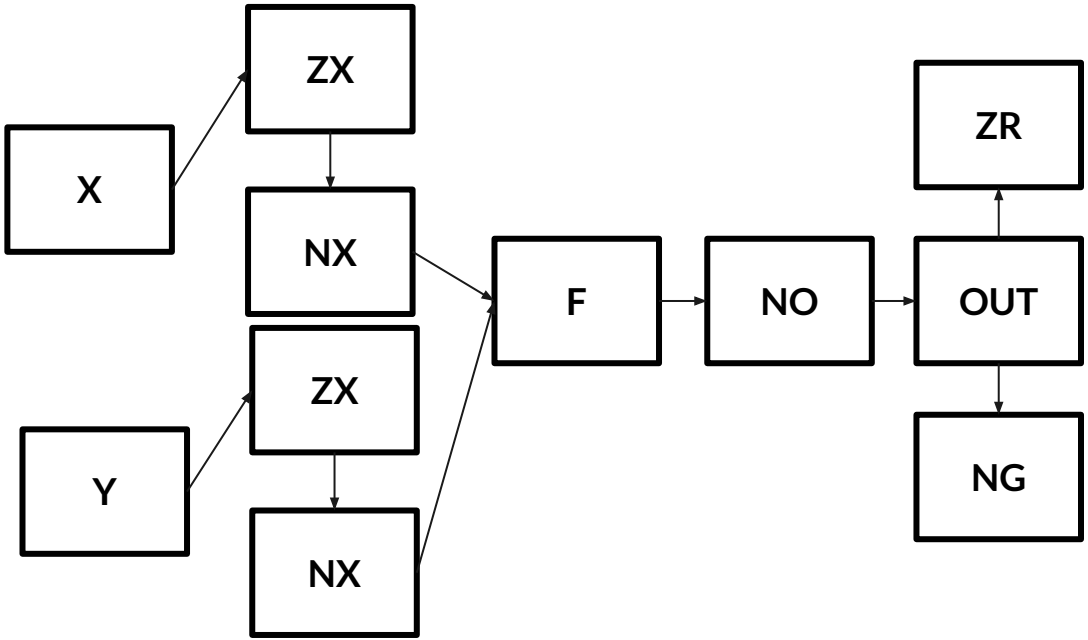
- Half-Adder
- Full-Adder
- Adder
- Incrementer



Project 2: Boolean Arithmetic

The Arithmetic Logic Unit (ALU)

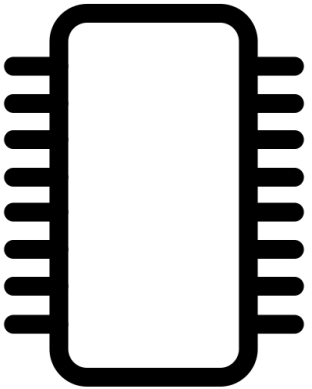




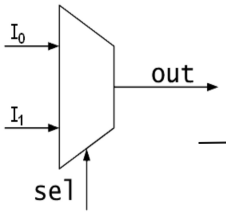
Project 3: Sequential Logic

Data-Flip-Flop

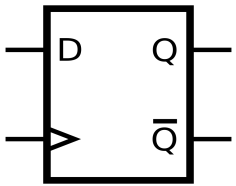
Registers



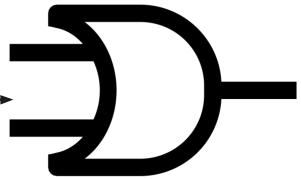
16-bit Register



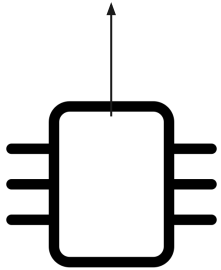
Multiplexor



D Flip-Flop



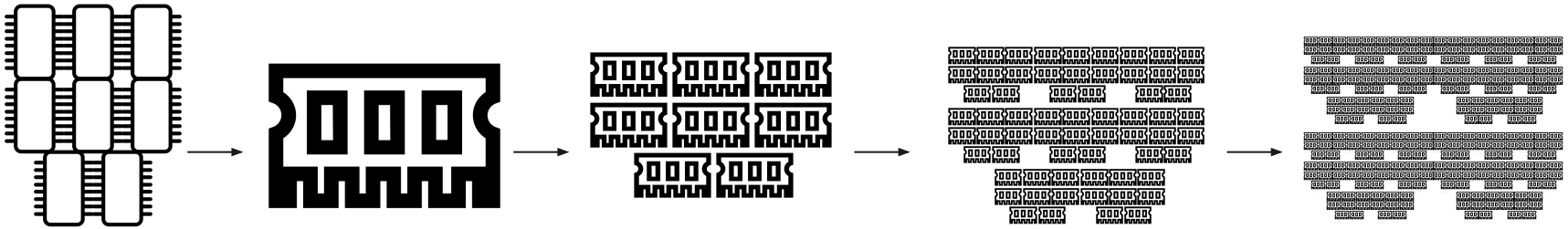
Or gate



Single-bit Register

Project 3: Sequential Logic

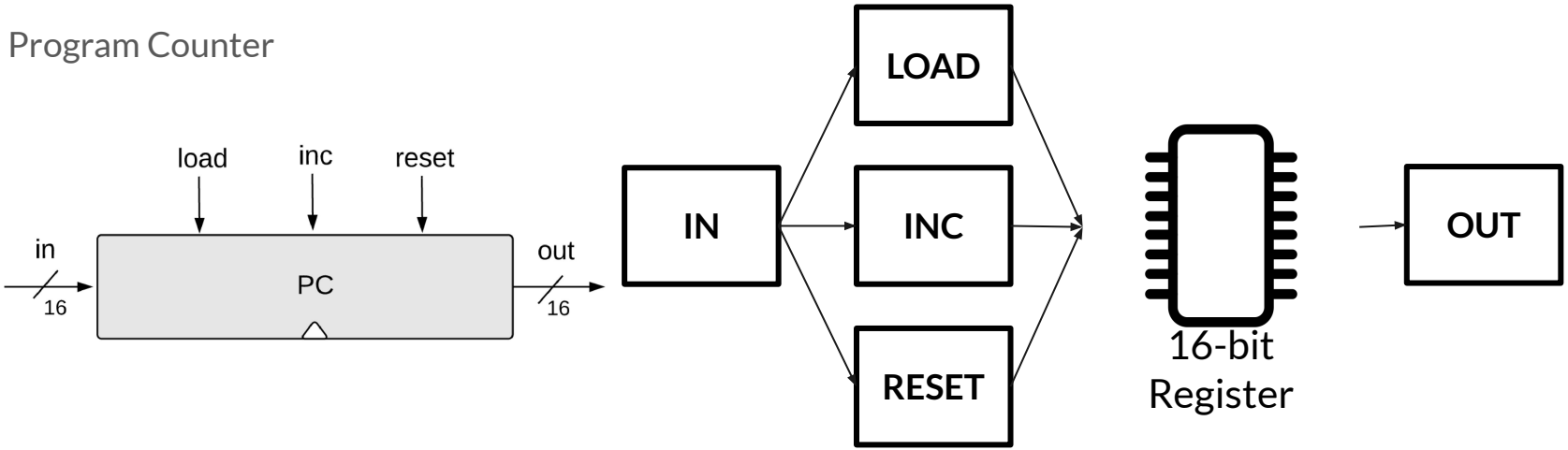
Memory



RAM8 RAM64 RAM512 RAM4K RAM16K

Project 3: Sequential Logic

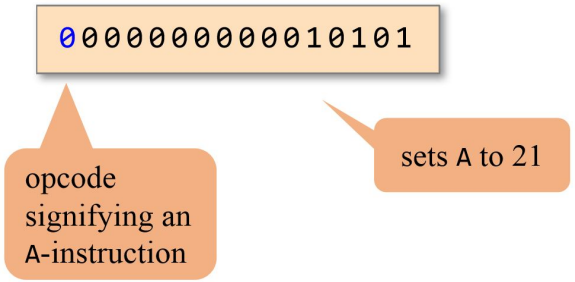
Program Counter



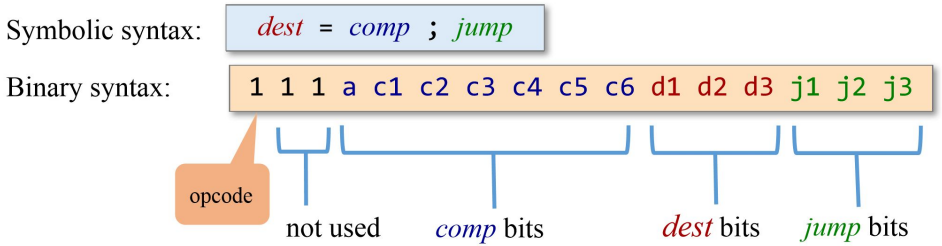
Project 4: Machine Language

Learn how to use Machine Language

Learn instruction components



A-instruction



C-instruction

Project 4: Machine Language

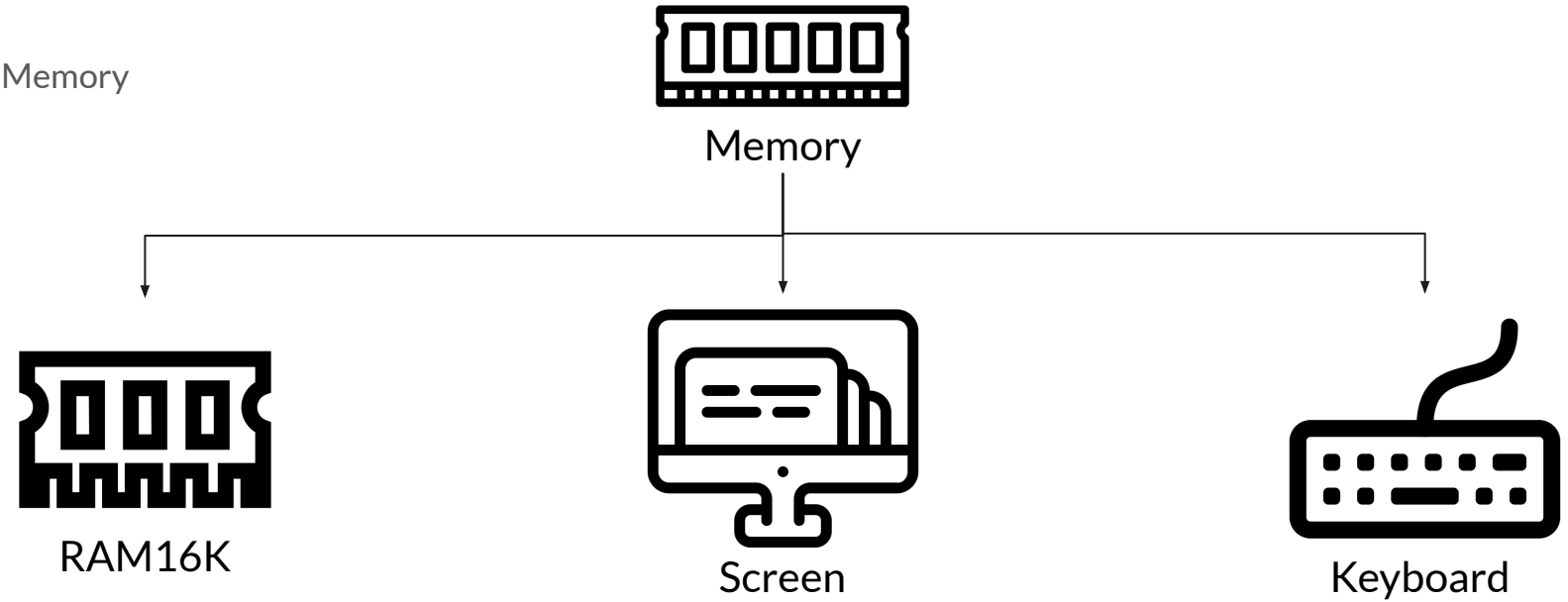
comp		c1	c2	c3	c4	c5	c6
0		1	0	1	0	1	0
1		1	1	1	1	1	1
-1		1	1	1	0	1	0
D		0	0	1	1	0	0
A	M	1	1	0	0	0	0
!D		0	0	1	1	0	1
!A	!M	1	1	0	0	0	1
-D		0	0	1	1	1	1
-A	-M	1	1	0	0	1	1
D+1		0	1	1	1	1	1
A+1	M+1	1	1	0	1	1	1
D-1		0	0	1	1	1	0
A-1	M-1	1	1	0	0	1	0
D+A	D+M	0	0	0	0	1	0
D-A	D-M	0	1	0	0	1	1
A-D	M-D	0	0	0	1	1	1
D&A	D&M	0	0	0	0	0	0
D|A	D|M	0	1	0	1	0	1
a==0	a==1						

dest	d1	d2	d3	effect: the value is stored in:
null	0	0	0	The value is not stored
M	0	0	1	RAM[A]
D	0	1	0	D register
MD	0	1	1	RAM[A] and D register
A	1	0	0	A register
AM	1	0	1	A register and RAM[A]
AD	1	1	0	A register and D register
AMD	1	1	1	A register, RAM[A], and D register

jump	j1	j2	j3	effect:
null	0	0	0	no jump
JGT	0	0	1	if out > 0 jump
JEQ	0	1	0	if out = 0 jump
JGE	0	1	1	if out ≥ 0 jump
JLT	1	0	0	if out < 0 jump
JNE	1	0	1	if out ≠ 0 jump
JLE	1	1	0	if out ≤ 0 jump
JMP	1	1	1	Unconditional jump

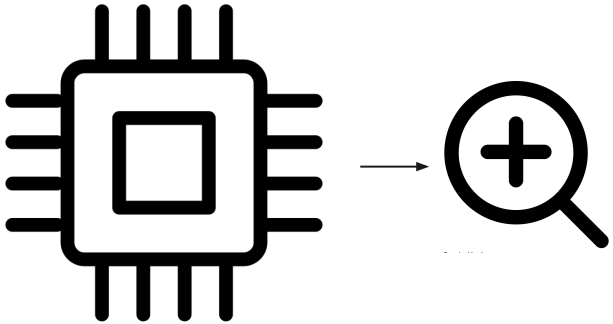
Project 5: Computer Architecture

Memory

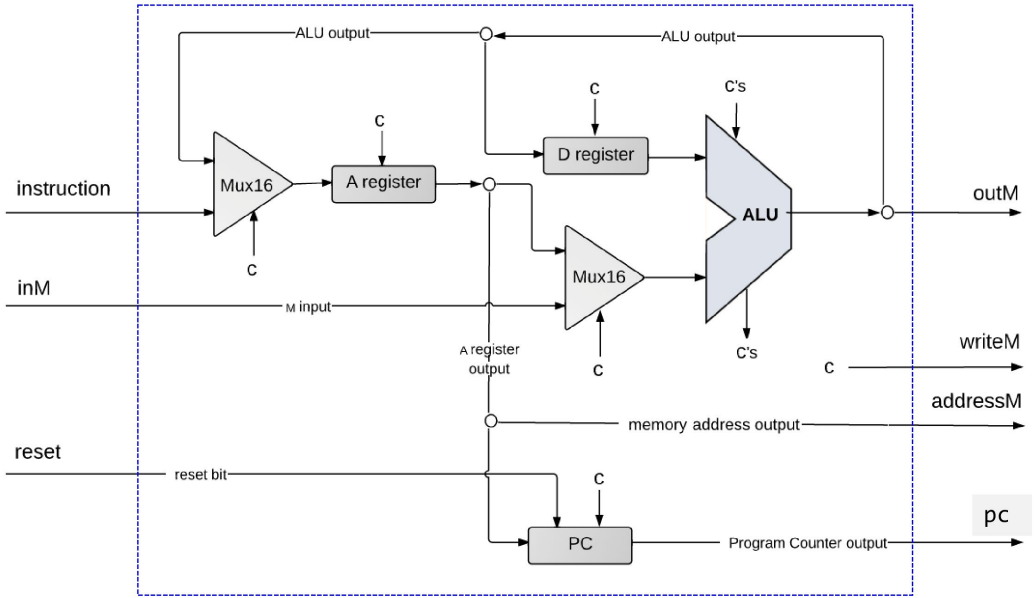


Project 5: Computer Architecture

CPU

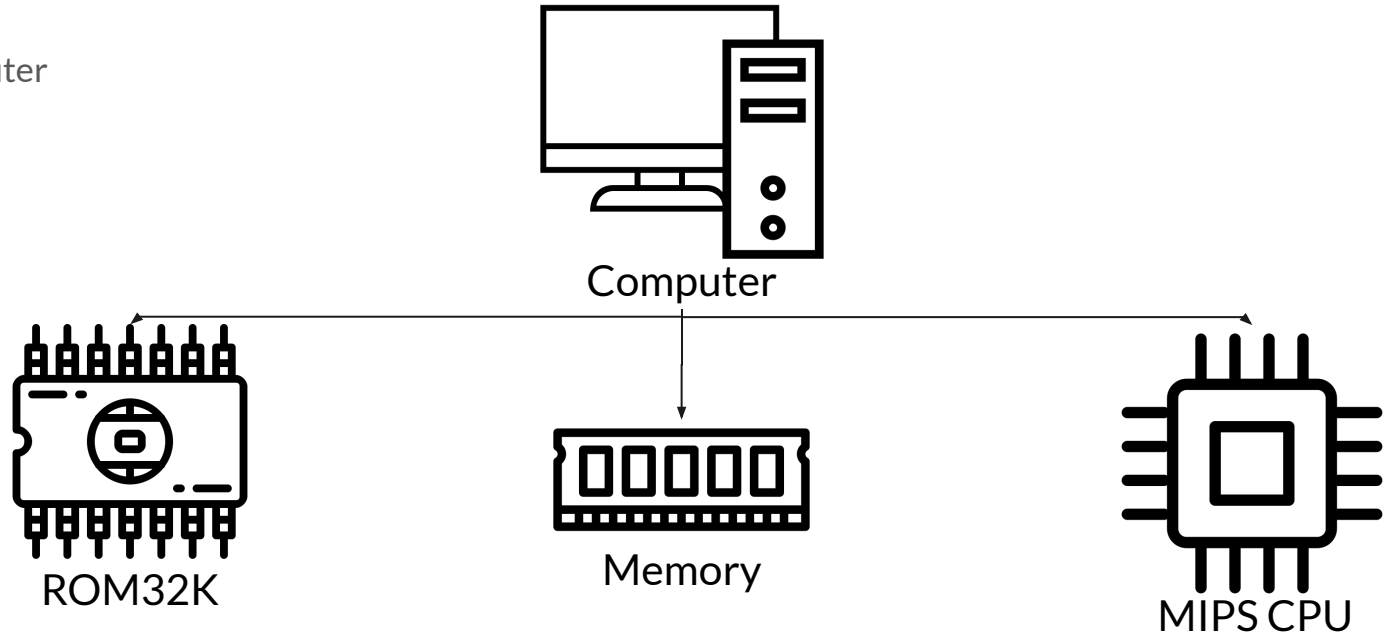


MIPS CPU



Project 5: Computer Architecture

Computer



Project 6: Assembler

Create Assembler using Java

- FileHelper.java
- Assembler.java

```java
import java.io.File;

public class FileHelper {

    /**
     * Delete comments(String after "//") from a String
     * @param strIn
     * @return
     */
    public static String noComments(String strIn){

        int position = strIn.indexOf("//");

        if (position != -1){

            strIn = strIn.substring(0, position);

        }

        return strIn;
    }

    /**
     * Delete spaces from a String
     * @param strIn
     * @return
     */
    public static String noSpaces(String strIn){
        String result = "";

        if (strIn.length() != 0){

            String[] segs = strIn.split(" ");

            for (String s: segs){
                result += s;
            }
        }

        return result;
```

FileHelper:
Manage file I/O

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.HashMap;
import java.util.Scanner;
import java.util.concurrent.ExecutionException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Assembler {
    public static HashMap<String,Integer> cMap = new HashMap<String, Integer>();
    public static HashMap<String,String> compAMap = new HashMap<String, String>();
    public static HashMap<String,String> compMMap = new HashMap<String, String>();
    public static HashMap<String,String> dstMap = new HashMap<String, String>();
    public static HashMap<String,String> jmpMap = new HashMap<String, String>();

    static {
        //put all predefined vars into a HashMap
        cMap.put("SP",0);cMap.put("LCL",1);cMap.put("ARG",2);cMap.put("THIS",3);
        cMap.put("THAT",4);cMap.put("R0",0);cMap.put("R1",1);cMap.put("R2",2);
        cMap.put("R3",3);cMap.put("R4",4);cMap.put("R5",5);cMap.put("R6",6);
        cMap.put("R7",7);cMap.put("R8",8);cMap.put("R9",9);cMap.put("R10",10);
        cMap.put("R11",11);cMap.put("R12",12);cMap.put("R13",13);cMap.put("R14",14);
        cMap.put("R15",15);cMap.put("SCREEN",16384);cMap.put("KBD",24576);

        //for c instructions comp=dst;jmp
        //put all comp posibilities with A into a HashMap,a=0
        compAMap.put("0","101010");compAMap.put("1","111111");compAMap.put("-1","111010");
        compAMap.put("D","001100");compAMap.put("A","110000");compAMap.put("!D","001101");
        compAMap.put("!A","110001");compAMap.put("-D","001111");compAMap.put("-A","110011");
        compAMap.put("D+1","011111");compAMap.put("A+1","110111");compAMap.put("D-1","001110");
        compAMap.put("A-1","110010");compAMap.put("D+A","000010");compAMap.put("D-A","010011");
        compAMap.put("A-D","000111");compAMap.put("D&A","000000");compAMap.put("D|A","010101");

        //put all comp posibilities with M into a HashMap,a=1
        compMMap.put("M","110000");compMMap.put("!M","110001");compMMap.put("-M","110011");
        compMMap.put("M+1","110111");compMMap.put("M-1","110010");compMMap.put("D+M","000010");
        compMMap.put("D-M","010011");compMMap.put("M-D","000111");compMMap.put("D&M","000000");
        compMMap.put("D|M","010101");
```

Assembler:
Convert assembly code into binary

Conclusion

The End

Thank for your listening!