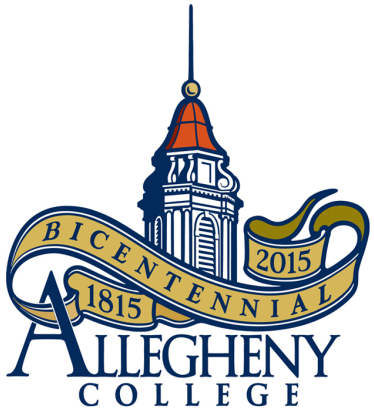
CS200 - Computer Organization

Assembly Language - 1

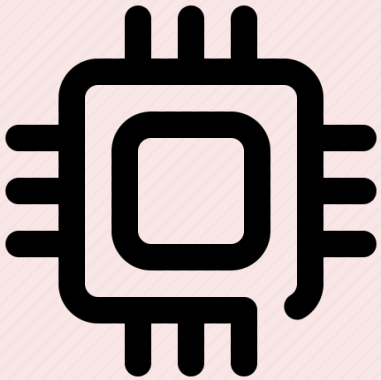
Aravind Mohan

Allegheny College

November 4, 2021



Motivation

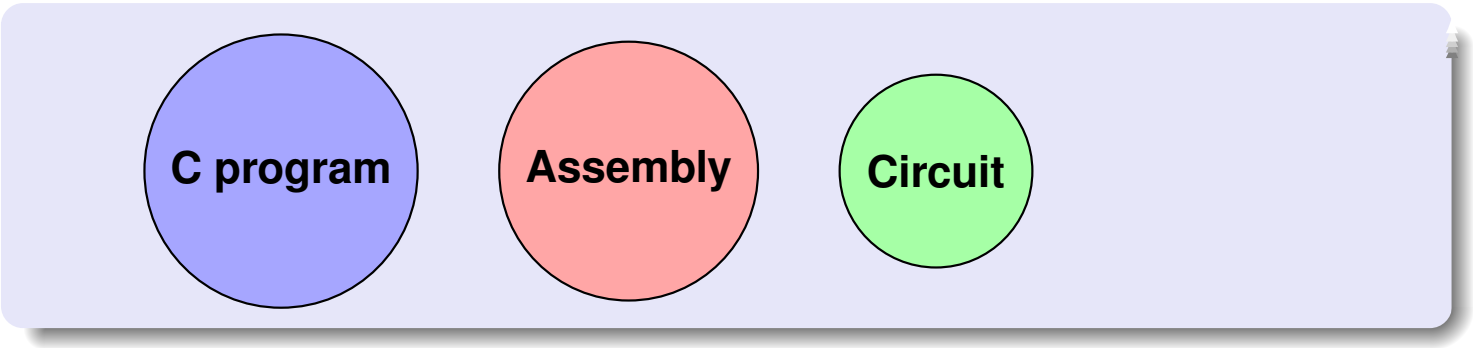


- To learn basics of Assembly Language programming.
- Realization of programming at the middle tier? That is at a layer, which is neither a hardware nor a software

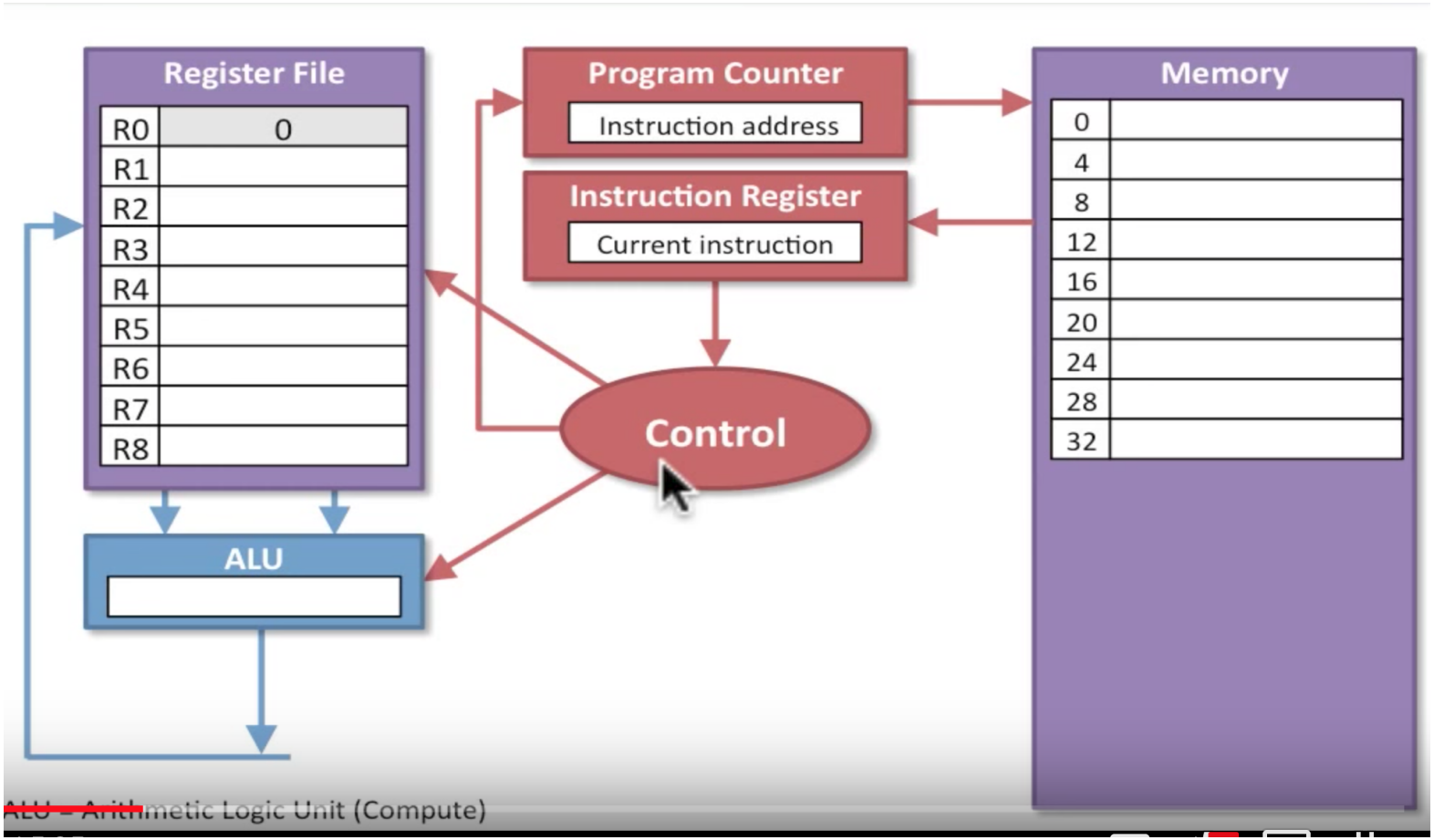
Assembly Language Programming

Low-level programming language in which there is a very strong correspondence between the instructions in the language and the architecture's machine code instructions.
Example: MIPS (wiki)

Computational Workflow



How instructions are executed?



How instructions are executed? (c'ntd)

- Program counter holds the instruction address
- Instructions are fetched from memory and placed into the instruction register.
- Control logic decodes the instruction and tells the ALU and register file what to do
- ALU executes the instruction and the results are placed into the register file
- The control logic updates the program counter to fetch the next instruction

Registers

- Registers are groups of flip-flops.
- The basic function of a register is to hold information in a digital system.

- Register **v0** = 1 to display Integer value
- Register **v0** = 4 to display String value
- Register **v0** = 11 to display Char value
- Register **v0** = 34 to display Hexa Decimal value

- Register **v0** = 5 to prompt Integer value
- Register **v0** = 8 to prompt String value

Registers c'ntd

register	assembly name	Comment
r0	$zero	Always 0
r1	$at	Reserved for assembler
r2-r3	$v0-$v1	Stores results
r4-r7	$a0-$a3	Stores arguments
r8-r15	$t0-$t7	Temporaries, not saved
r16-r23	$s0-$s7	Contents saved for later use
r24-r25	$t8-$t9	More temporaries, not saved
r26-r27	$k0-$k1	Reserved by operating system
r28	$gp	Global pointer
r29	$sp	Stack pointer
r30	$fp	Frame pointer
r31	$ra	Return address

MIPS Hello World

```c
#include <stdio.h>
int main(){
   printf("Hello World!\n");
   return 0;
}
```

```
.data
myMessage: .asciiz "Hello World \n"
.text
la $a0, myMessage
li $v0, 4
syscall
```

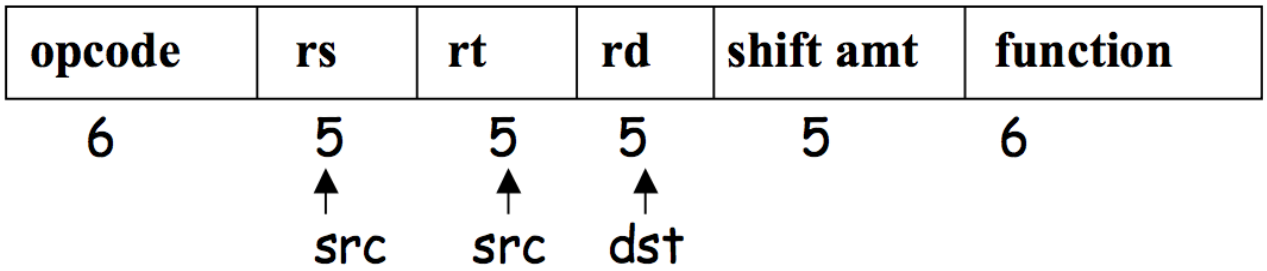
java -jar mars.jar hello.asm

Some MIPS Instructions

- add, addi, sub, subi, andi, sll, srl, beq, bne, bge, move, la, li, lw, lb, lh, sw, sh, sb, j, jal, syscall

MIPS Format

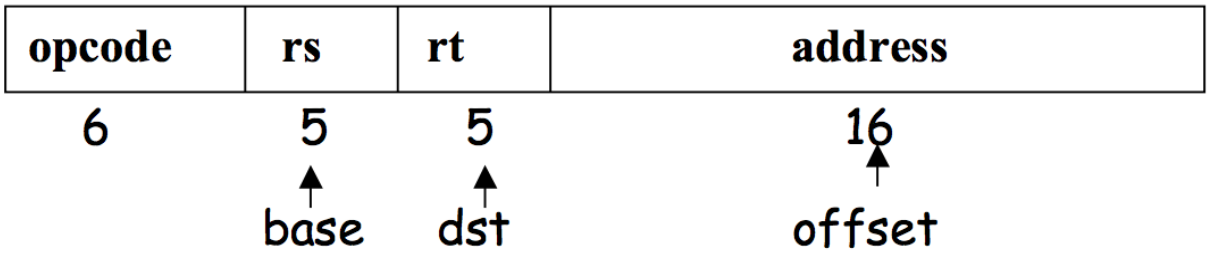
R-type format



Used by **add**, **sub** etc.

MIPS Format

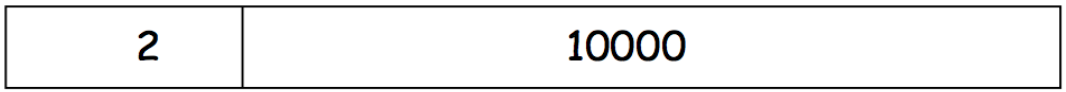
I-type format



Used by lw (load word), sw (store word) etc

MIPS Format

J 10000 is represented as

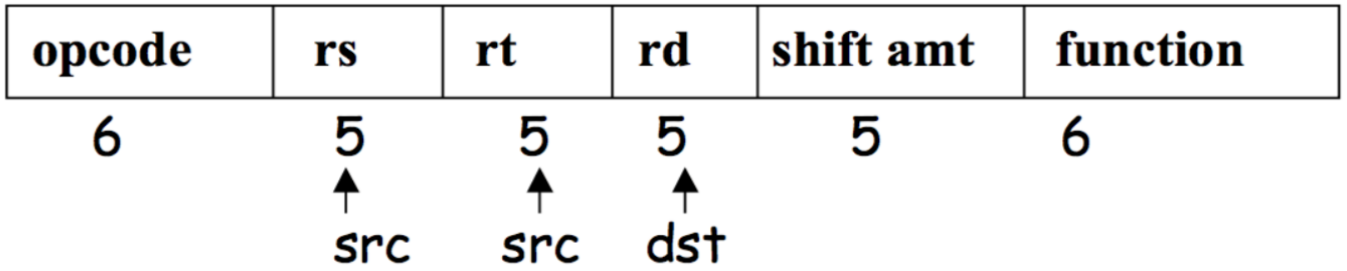


6-bits 26 bits

This is the J-type format of MIPS instructions.

Instruction Decoding

- Decode the instruction **add $t2, $t0, $t1**:



- Add = opcode 0
- $t2 = register 10
- $t0 = register 8
- $t1 = register 9
- No shift amount
- Add = function 32

Instruction Decoding

- Add = opcode 0 => 000000
- $t2 = register 10 => 01010
- $t0 = register 8 => 01000
- $t1 = register 9 => 01001
- No shift amount => 00000
- Add = function 32 => 100000

Instruction Decoding

- 000000 01000 01001 01010 00000 100000
- 0000 0001 0000 1001 0101 0000 0010 0000
- 0x01095020

Instruction Decoding

- Can we see this in Mars?

java -jar mars.jar a dump .text HexText hexcode.txt add.asm

Practice

- Try out:
- Decode the instruction **add $t3, $t1, $t2**:
- Decode the instruction **sub $t1, $t2, $t3**:

  sub op code is 000000 and function code is 34

  register values are available in slide 8

Reading Assignment

- Computer Organization and Design by Patterson and Hennesssy - Chapter 04 - [4.2 - 4.4];

Questions

Do you have any questions from this class discussion?