CS200 - Computer Organization

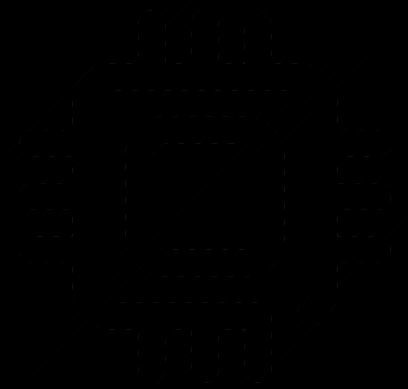
Assembly Language - 1

Aravind Mohan

Allegheny College

November 8, 2022

Motivation

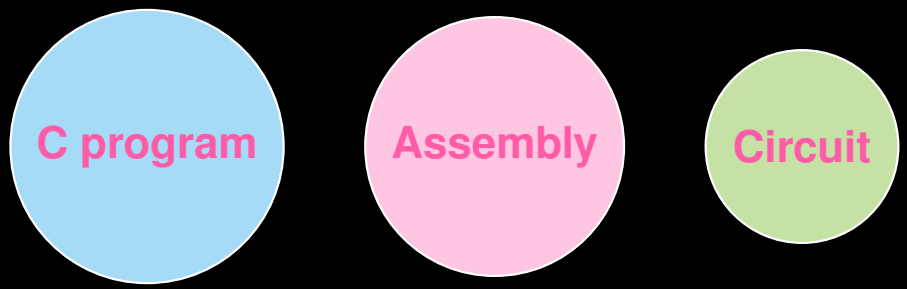


- To learn basics of Assembly Language programming.
- Realization of programming at the middle tier? That is at a layer, which is neither a hardware nor a software

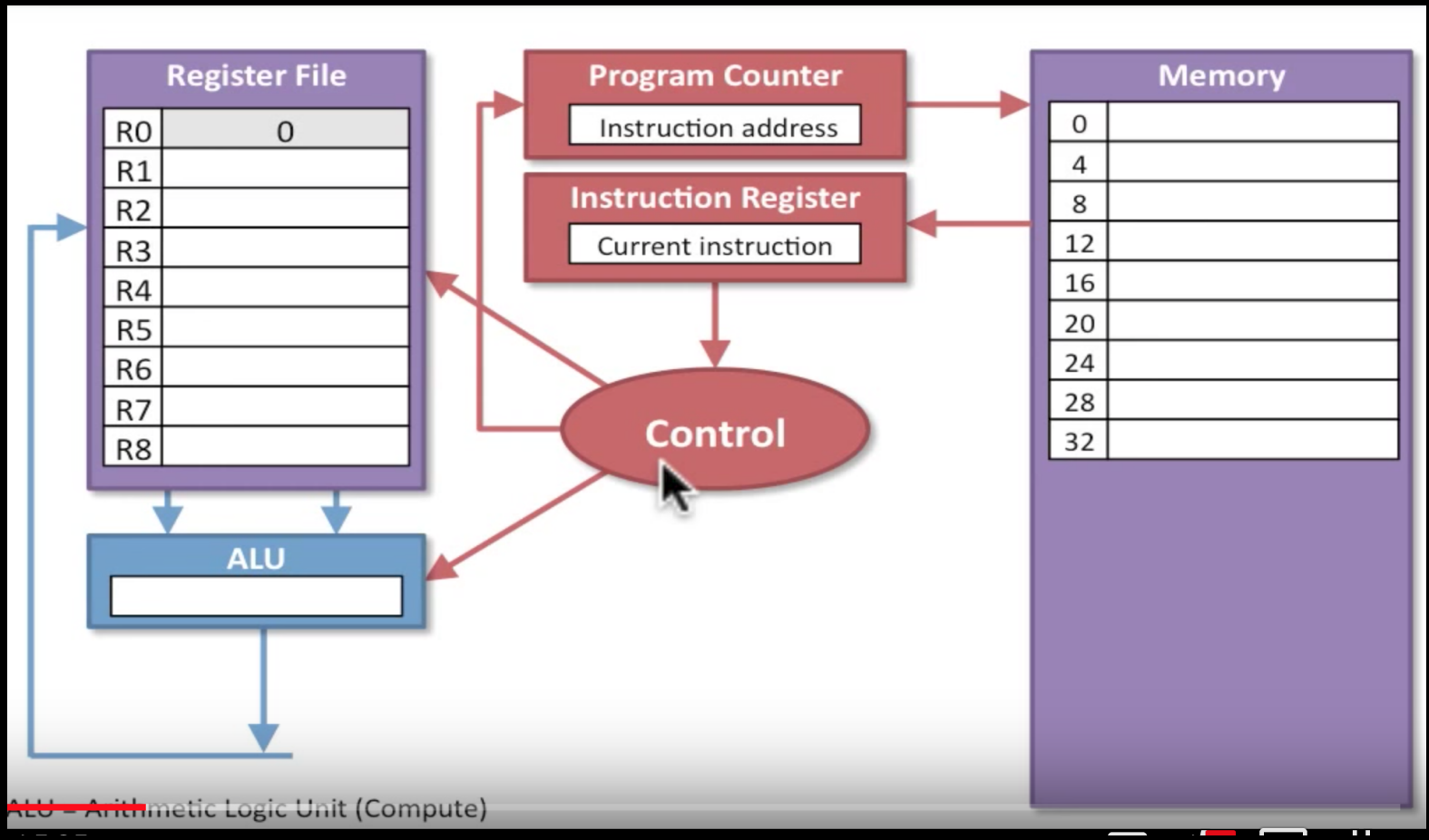
Assembly Language Programming

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
```

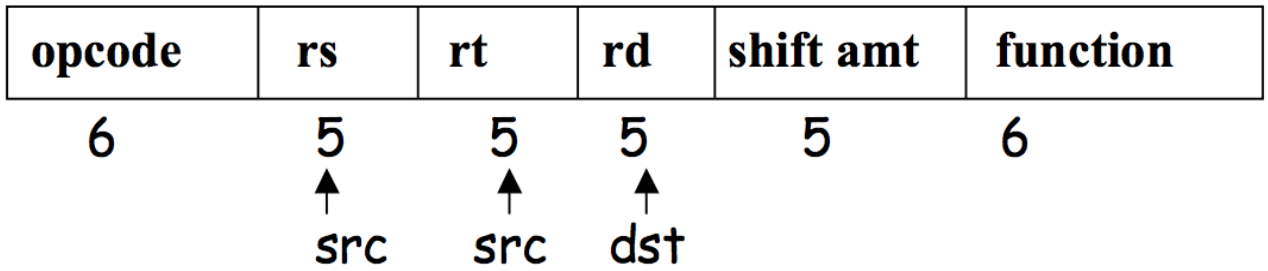
java -jar mars.jar hello.asm

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

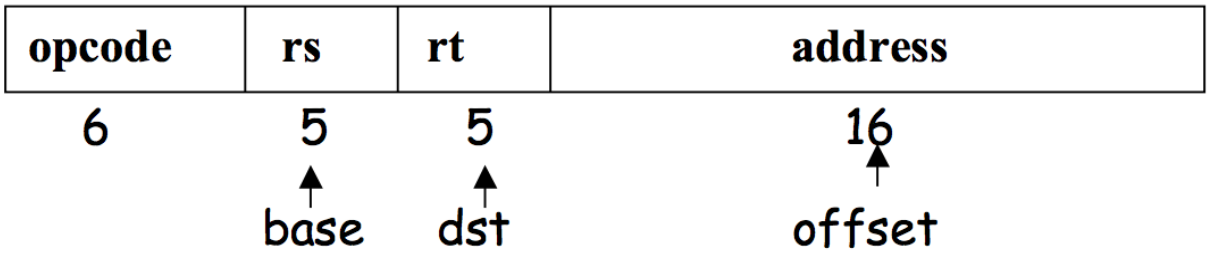
R-type format



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

MIPS Format

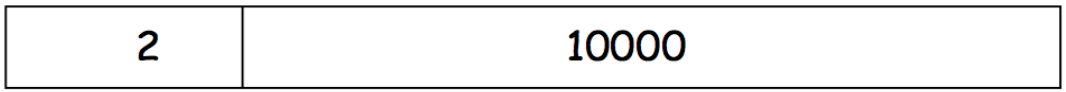
I-type format



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

MIPS Format

J 10000 is represented as

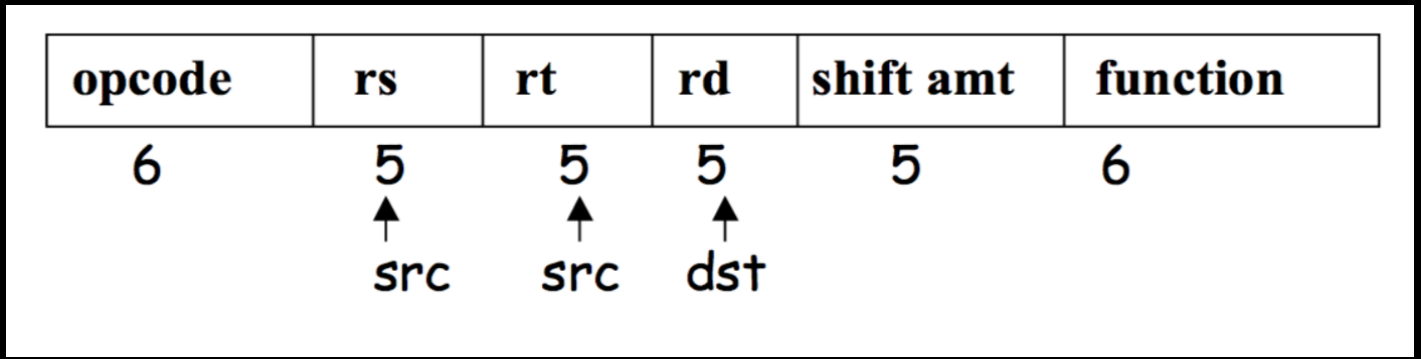


6-bits 26 bits

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

Calculator Implementation in ASM

- add, sub, mul, div

see calculator.asm in the repo

Conditional Logic

- branch if greater than **bge**
- branch if less than **ble**
- branch if equal **beq**
- branch if notequal **bne**

Conditional Logic (c'ntd)

```
#include <stdio.h>
int main(){
  int temp = 0;
  printf("Enter the current temperature:\n");
  scanf("%d", &temp);
  if (temp >= 65)
    printf("hot\n");
  else
    printf("cold\n");
  return 0;
}
```

Conditional Logic (c'ntd)

```
.data
prompt:   .asciiz     "\nEnter the current temperature:\n"
hot:      .asciiz     "hot\n"
cold:     .asciiz     "cold\n"
.text
# prompt the current temp
li $v0, 4
la $a0, prompt
syscall
# read the current temp
li $v0, 5
syscall
move $t0, $v0
# load value to register
li $t1, 65

main:
  bge $t0,$t1,rule1
  beq $t0,$t1,rule1
  ble $t0,$t1,rule2
```

Conditional Logic (c'ntd)

```
rule1:
   li $v0, 4
   la $a0, hot
   syscall
   j exit

rule2:
   li $v0, 4
   la $a0, cold
   syscall
   j exit

exit:
   li $v0, 10
   syscall
```

Conditional Logic (c'ntd)

- What if we introduce a new rule?

  $\alpha >= 50$ and $\alpha < 65$

  then print "Mild"

  $\alpha < 50$

  then print "Cold"

see temp.c and temp.asm in the repo

Iterative Logic

- jump **j**

Counter

```c
#include <stdio.h>
int main(){
  for (int i = 1; i <= 5; i++)
    printf("%d\n",i);
  return 0;
}
```

see counter.c and counter.asm in the repo

Counter (c'ntd)

```
.data
break:   .asciiz      "\n"
.text
li $t1, 1
li $t2, 1
li $t3, 5
counter:
     li $v0, 1
     la $a0, 0($t1)
     syscall
     li $v0, 4
     la $a0, break
     syscall
     add $t1,$t1,$t2
     beq $t1,$t3,exit
     j counter
exit:
     li $v0, 10
     syscall
```

Counter (c'ntd)

- How do you build a counter that prints in descending order?

Counter (c'ntd)

- Print numbers from 0 to 10, in increment of 2.
- Print numbers from 10 to 0, in decrement of 2.

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?