Assembly Language Programming

CS2400

Fall 2020

MIPS

MIPS (Microprocessor without Interlocked Pipelined Stages)

MIPS 32-bit microprocessor

MIPS use RISC (Others are ARM [Advanced RISC], X86 [CISC])

RISC – Reduced Instruction Set Computer Architecture

MIPS Instruction Set Architecture (ISA)

High level code: **a= b + c**

Assembly code: **add a, b, c** # a is the sum of b and c

Machine code: 00000010001100100100000000100000 (32 bit)

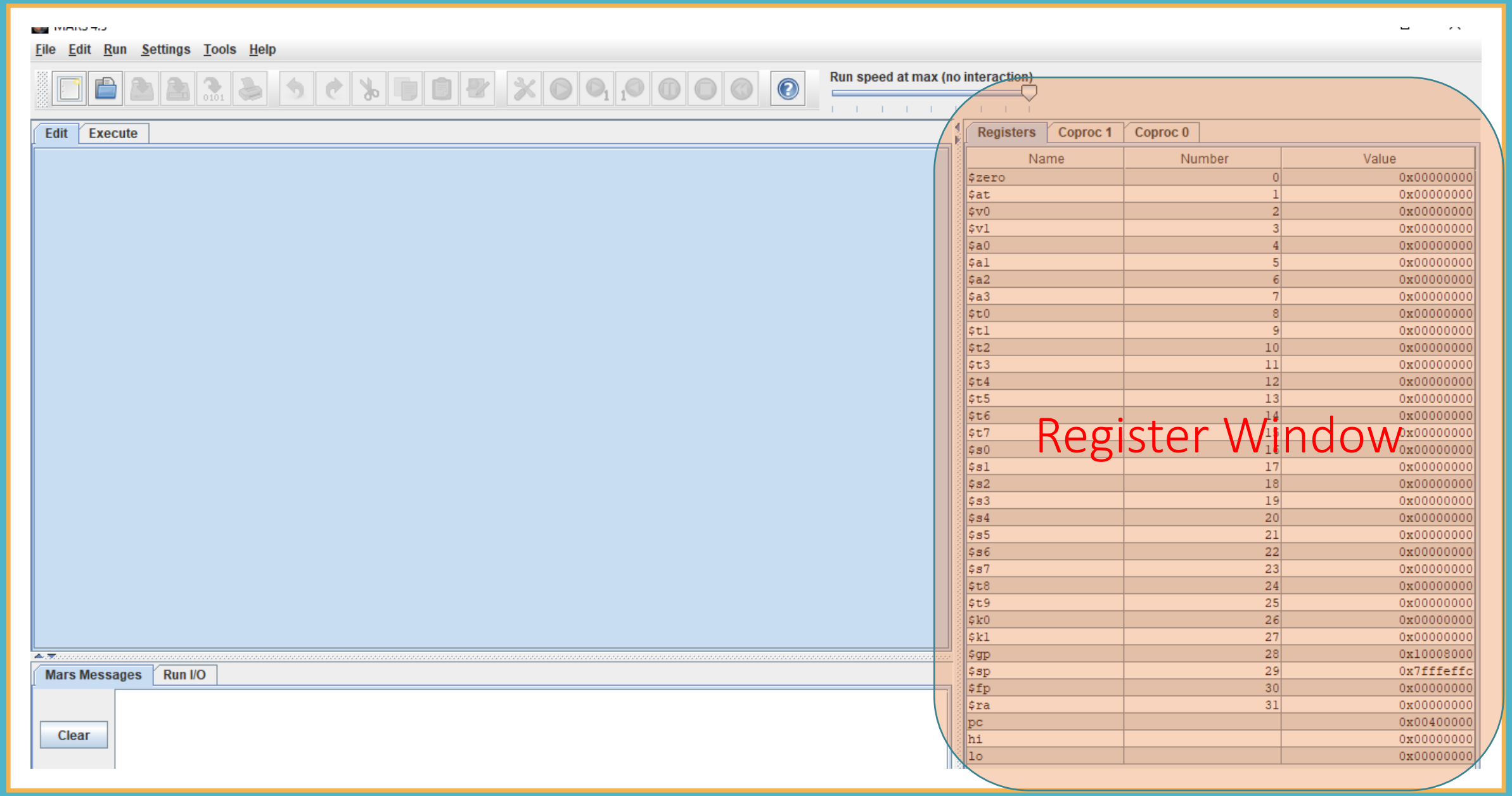
Registers

The MIPS central processing unit contains 32 general purpose registers.

Registers are 32 bit.

Register $0 always contains the hardwired value 0.

MIPS has established a set of conventions as to how registers should be used.


File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Mars Messages Run I/O
Clear
Registers Coproc 1 Coproc 0
Name Number Value
$zero 0 0x00000000
$at 1 0x00000000
$v0 2 0x00000000
$v1 3 0x00000000
$a0 4 0x00000000
$a1 5 0x00000000
$a2 6 0x00000000
$a3 7 0x00000000
$t0 8 0x00000000
$t1 9 0x00000000
$t2 10 0x00000000
$t3 11 0x00000000
$t4 12 0x00000000
$t5 13 0x00000000
$t6 14 0x00000000
$t7 15 0x00000000
$s0 16 0x00000000
$s1 17 0x00000000
$s2 18 0x00000000
$s3 19 0x00000000
$s4 20 0x00000000
$s5 21 0x00000000
$s6 22 0x00000000
$s7 23 0x00000000
$t8 24 0x00000000
$t9 25 0x00000000
$k0 26 0x00000000
$k1 27 0x00000000
$gp 28 0x10008000
$sp 29 0x7fffeffc
$fp 30 0x00000000
$ra 31 0x00000000
pc 0x00400000
hi 0x00000000
lo 0x00000000
Register Window

Registers

Registers $at (1), $k0 (26), and $k1 (27) are reserved for use by the assembler and operating system.

Registers $a0-$a3 (4-7) are used to pass the first four arguments to routines (remaining arguments are passed on the stack).

Registers $v0 and $v1 (2, 3) are used to return values from functions.

Registers $t0-$t9 (8-15, 24, 25) are caller-saved registers used for temporary quantities that do not need to be preserved across calls.

Registers $s0-$s7 (16-23) are callee-saved registers that hold long-lived values that should be preserved across calls.

Register $sp (29) is the stack pointer, which points to the last location in use on the stack.

Register $fp (30) is the frame pointer.

Register $ra (31) is written with the return address for a call by the jal instruction.

Register $gp (28) is a global pointer that points into the middle of a 64K block of memory in the heap that holds constants and global variables. The objects in this heap can be quickly accessed with a single load or store instruction.

When you hear, you forget;

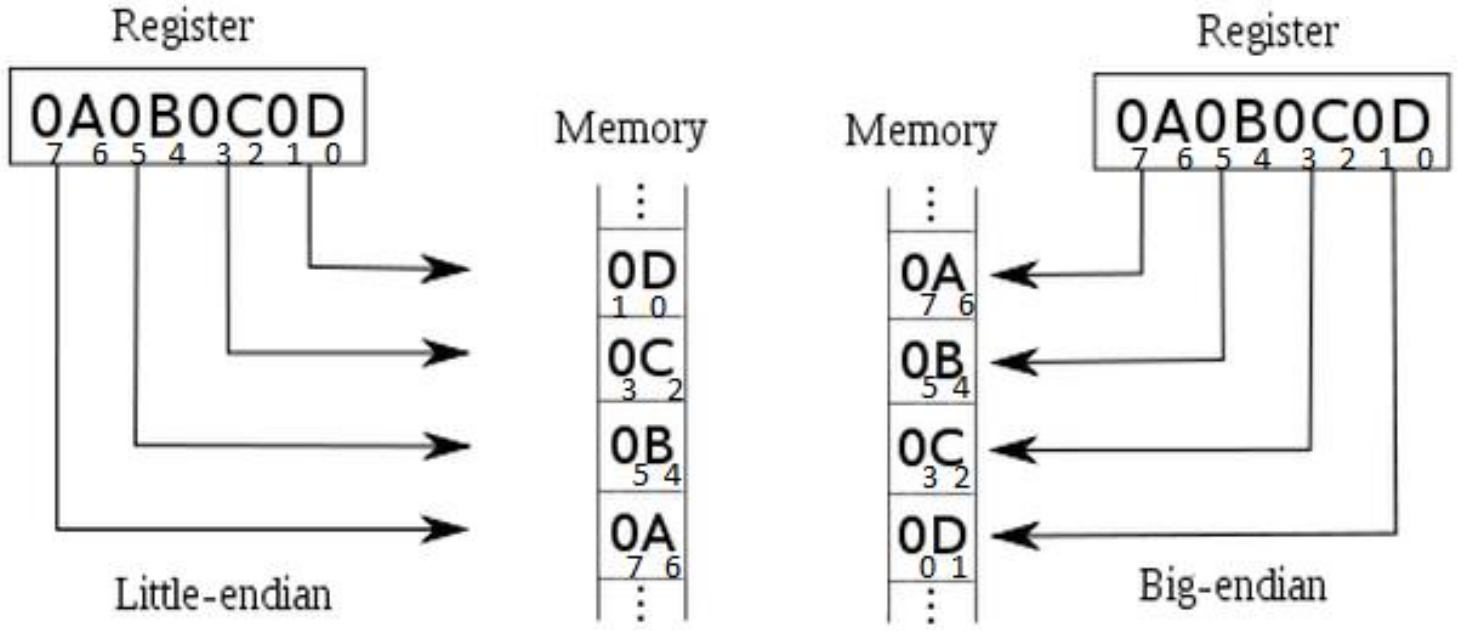
When you see, you remember;

When you do you understand.

Byte Order

Byte Order

Processors can number the bytes within a word to make the byte with the lowest number either the leftmost or rightmost one. The convention used by a machine is its *byte order*. MIPS processors can operate with either *big-endian* byte order or *little-endian* byte order.



Addressing Modes

MIPS is a load/store architecture, which means that only load and store instructions access memory.

Computation instructions operate only on values in registers.

Memory Model

- 32-bit address Format

- The addresses of MIPS main memory range from 0x00000000 to 0xFFFFFFFF.

- **Load:** a bit pattern starting at a designated address in memory is copied into a register inside the processor.
- **Store:** a bit pattern is copied from a processor register to memory at a designated address.

MARS IDE

Functions

- Text editor to writing code
- Loads and executes MIPS assembly language files (.asm or .s)
- Debugger (breakpoints, single-stepping)
- Provides Console for input/output

INTERFACE

Registers window	Text segment window	Data segment window	Messages window	Console interface
• values of all registers Int/FP	• user and kernel instructions • Addresses, code in binary and assembly	• User data, stack, kernel data	• messages (include error messages)	• Input/output

My First Program

label

directives

```
#data section
.data
    myMessage: .asciiz "Welcome CS2400\n"

#code section
.text
    li $v0, 4                  # system call code for print string; li is load immediate
    la $a0, myMessage          # address of string to print; la is load address
    syscall                    #print the string (invoke system call 4 defined above)
```

Classwork 01

Write an assembly code to print the last digit of your #900 number and your name.

15 min and then we will discuss,

Classwork 02

Modify the program to accept the name and last digit of #900 number from the user and Display the Name and number as given below.

Example output “Ranjidha 6 “ # where 6 is Ranjidha’s last digit in the #900

To Do as of 08/24/2020

- Exit Quiz 2

  Submit the code you did in Classwork 2 - Modify the program to accept the name and last number of #900 number from the user and Display the Name and number as given below.

  Example output “Ranjidha 6 “

- Read chapter1 section“ Performance “ from Material A before next class.