Mangesh Raut mbr63

Assignment Week2

CS503 System Basics

Part1: Concepts

Q1:

a)

Ans: In computer programming, assembly language, is any 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.

b)

Ans: MIPS assembly language simply refers to the assembly language of the MIPS processor. The term MIPS is an acronym for Microprocessor without Interlocked Pipeline Stages. It is a reduced instruction set architecture developed by an organization called MIPS Technologies.

c)

Ans: SPIM is a MIPS processor simulator, designed to run assembly language code for this architecture. The program simulates R2000 and R3000 processors and was written by James R. Larus while a professor at the University of Wisconsin–Madison.

d)

Ans: Assembly language was created as an exact shorthand for machine level coding, so that you wouldn't have to count 0s and 1s all day. It works the same as machine level code: with instructions and operands. Well, there's only one language we will ever need, which is called "machine language" or "machine code". It looks like this: 010111011100. This is the only language your computer can speak directly. It is the language a CPU speaks (and technically, different types of CPUs speak different versions). It also sucks to look at and try to understand.

Q2:

a)

1. Ans: .data signals the start of the data section. The data section is used for declaring initialized data or constants. This data does not change at runtime. You can declare various constant values, file names, or buffer size, etc., in this section.

2. Ans: .word is 32 bit of memory. A word is a fixed-sized datum handled as a unit by the instruction set or the hardware of the processor. The number of bits or digits in a word (the word size, word width, or word length) is an important characteristic of any specific processor design or computer architecture.

3. Ans: asciiz is a compile time assembler directive. You can use .asciiz to allocate space for the user input if you want, but you still need to fill it at runtime, for example by reading characters in a loop and storing them using sb or using read string or similar system call if you have one.

4. Ans: Actual code are put into .text section. .text tells the assembler to switch to the text segment (where code goes). This symbol is of type text. It is used to label the beginning of a .text section in the program being assembled.

5. Ans: These are assembler directives. We define the data section for declaring the initialized data we define x,y,z. to 0 using .word is 32 bit of memory. nl initialized to locate space. As “\n” also indicate the new line. text tells the assembler to switch to the text segment (where code goes).

b)

1. Ans: msg1: .asciiz "Enter x value "

li $v0,5 # read\_int syscall code = 5

syscall

move $t0,$v0 # syscall results returned in $v0

2. Ans: load value 5 into register v0

3. Ans:

We have to change in main to print the message then in loop we have to use li, $v0, 4 syscall to read the value of x and y. we also have to look on .text and .data.

.data

msg1: .asciiz "Enter x value "

msg2: .asciiz "Enter y value "

c)

1. Ans: li $v0, 1 # Print z

lw $a0, 0($t0)

syscall

2. Ans: li $v0,4 # code 4 == print string

la $a0,z # $a0 == address of the string

syscall # Ask the operating system to

# perform the service.

. . . .

.data

z: .asciiz "Hello SPIM!\n"

Part 2:

Graphical user interface

Description automatically generated

Part 3:

add.asm

Graphical user interface

Description automatically generated

Countdown.asm

Graphical user interface

Description automatically generated

Part 4:Graphical user interface

Description automatically generated