~~Control~~

See Spreadsheet

The `lui` instruction uses a register (usually `$1` which is a register reserved for the assembler) and a constant. In the table below enter the constant you find with the first `lui` in your program, both in decimal and in 16 bit hexadecimal format.

Decimal constant	16 bit hexadecimal
4097	1001

Reload *lab2_1.asm* and set a breakpoint at the address of that `lui` instruction. Run, and the program will stop at that breakpoint. Write down the contents of the register used by the instruction. Execute that instruction (`step 1`) and write down the content of the same register.

Register	Before executing `lui`	After executing `lui`
$1	0	10010000

Q 1:

Assume you want to load the constant `0xabcd0000` in register **`$t0`**. What native instruction(s) should be executed?

lui $1, 43981
ori $8, $1, 0

Step 4

What is the content of register **`$t2`** after the instruction `la $t2, var2` has been executed? Note that this

10010004

synthetic instruction contains a `lui` followed by an immediate instruction.

Q 2:

What does the immediate instruction do in this case?

it ors 10010000 with 00000004

Q 3:

Assume you want to load the constant `0xabcd00ef` in register **`$t0`**. What native instruction(s) should be

executed?

w: $1,439.81 or: $8,$1,239

Variable name	Initial value
var1	x0000
var2	x0001
var3	xabcd
var4	x1000020

y for four variables called *var1* thro
e those provided by your lab instruct
emory for two variables called *first*
etter of your first name and the initia

values of variables in memory: the n
of *var2* will be the initial value of *va*
the initial value of *var1*. *first* and *la*
8 if you need to
tion set
has a comment indicating what the

of each variable in your program fro
ce in bytes between the beginning of
nt command to see what is stored i

Variable	Displacement
var1	x10003
var2	x10004
var3	x10006
var4	x10008
first	x1001D

Laboratory 2: Postlab

Date _____ Section _____

Name _____

Addressing in MIPS (continued)

In this exercise you will continue exploring addressing in MIPS. You are also required to detail the memory model SPIM implements.

Step 1

As you could see during the inlab exercise, one can not run on the bare machine code that uses the extended instruction set.

Based on *lab2.2.asm* create a new program that will do the same thing but will be able to run on the bare machine. Save this as *lab2.3.asm*. Optimize your code as much as possible.

Q 1:

What is the number of instructions executed? Count only instructions between the label 'main' and the last instruction executed from your program.

Step 3

Run the program and make sure it works properly.

Q 1:

What is the number of instructions executed? Count only instructions between the label 'main' and the last instruction executed from your program.

Instruction Count = 47

Step 4

Start the simulator using the `-bare` command line option. This will make SPIM simulate a bare MIPS processor.

Load *lab2.2.asm*. What is the reason you get error messages?

Because -bare doesn't use the extended instruction set

Q 2:

What are the displacements of *ext1* and *ext2* from the global pointer (`$gp`) value?

Variable	Displacement (decimal)	Displacement (hexadecimal)
ext1	-32768	8000
ext2	-32764	8004

Q 3:

What exactly are the addresses where variables are stored in memory?

Variable	Address (hexadecimal)
var1	10010000
var2	10010002
ext1	10010000
ext2	10000000

Q 4:

How many native instructions are needed for each of the following memory accesses?

Memory Access	Native Instructions