040003 H			Assertion
		× -	
		\	
	See Se	ocen) sheef	
	-		
	7		

The 111 instruction uses a register (usually \$1 which is a register reserved.)

In the table below enter the constant you find with the first 111 in your program, both in decimal and in 16 bit hexadecimal format.

Decimal constant	16 bit hexadecimal
9097	1001

Reload lab2. Lasm and set a breakpoint at the address of that 1u1 instruction. Run, and the program will stop at that breakpoint. Write down the contents of the register used by the instruction. Execute that instruction (at ep 1) and write down the content of the same register.

Register	Before executing lui	After executing lui
31	0	10010000

#### Q 1:

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

10;	\$1,	439	81	
00	18,	41,	0	

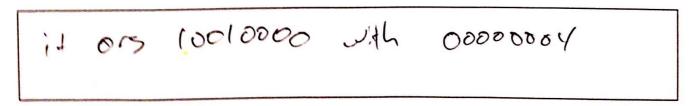
#### Step 4

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

synthetic instruction contains a lui followed by an immediate instruction.

## Q 2:

What does the immediate instruction do in this case?



#### Q 3:

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

# **Scanned by CamScanner**

executed?

10: \$1,43981 01:\$8,\$1,239

Variable name	Initial value
varl	x 0000
var2	x 2001
var3	xated
var4	x1000000

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 value of var1. 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 from ce in bytes between the beginning of nt command to see what is stored in

Variable	Displacement
varl	×10003
var2	Y0001X
var3	×1000 6
var4	×10008
first	Scanned by

**CamScanner** 

	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.

#### Q1:

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 = 8 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 fle extended instruction set

# Q 2:

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

Variable	Displacement (decimal)	Displacement (hexadecimal)
extl	-37768	\$000
१क्षे	-327624	8004

# Q 3:

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

Variable	Address (hexadecimal)
varl	1001000
var2	10010002
extl	000000
ext2	100000000

## Q 4:

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

Memory Access	Native Instructions
Wiemory Meeess	rative instructions

# **Scanned by CamScanner**