Low-Level Programming

Names: Teddy (Theodore) Clapp, Octavia Clapp

Brief Introduction (definitions)
Abstraction is a more generic supertype where concrete subtypes tend to have many similari tenacities to the abstracted. MIPS is a language where one can write assembly that is such low level, that hardware control is high. SPIM is the physical unit, for processing, that MIPS is intended to be written for.

QUESTION:
In this project you are asked to use the term abstraction – please define abstraction from the perspective of computer science.

ANSWER:
Abstraction can be used to describe object oriented hierarchies. It is common for an abstraction to be a more generic supertype of a more concrete entity.

QUESTION:
The registers also have various labels, such as **t0** through **t7**, **v0**, and **a0** (all of which you used in the code from Part 1). You can see these labels in brackets (e.g., **[t0]**) in the left window. These labels have a specific meaning. When writing code, what is the benefit of using these labels instead of the register numbers?
Refer to the concept of *abstraction* in your answer.

ANSWER:
This way you don't have to know the exact number, but rather, you get use the label, and by doing so, you'll be getting the number. This is a form of abstraction because one can grab a generically named name while actually obtaining a concrete number.

QUESTION:
Even though '19' was loaded into $t0, SPIM shows this register to have the value '13' at the top section of the screen. This is not a mistake; '19' was loaded into $t0. How is SPIM displaying the value of $t0 differently such that you see '13' instead of '19'?

ANSWER:
SPIM converts base ten (decimal) to base 16 (hex).

QUESTION:
The instruction 'mul $t2, $t0, $t1' should now be highlighted. In the space below, hypothesize what the processor will do. Then hit the F10 key once.

ANSWER:
I predict that the values at t1 and t0 will be multiplied, and their value will be stored in t2

What value landed in $t2? Write your answer in hexadecimal.

ANSWER:
0x2BF

QUESTION:
How is the value in $t2 related to the values in $t0 and $t1?

ANSWER:
$t0 and $t1 are factors of t2.

You should be at the line marked 0x00400030 with the instruction ‘li $t3, 42’ highlighted. Press F10 to execute this instruction. Examine the value in $t3. What did the instruction do?
ANSWER:
This loaded the integer 42 into t3.

QUESTION:
The current instruction to be executed should be ‘add $t4, $t2, $t3’. Press F10 to execute this instruction. Examine the values in $t2, $t3, and $t4. What did the instruction do?

ANSWER:
This took values at t2 and t3 and performed addition whilst storing the sum into t4.

QUESTION:
You should be at instruction 0x00400038. The instruction on this line is ‘addi $t5, $t4, 0’. Press F10 and note that $t5 equals $t4. Mathematically, what did this instruction do with each of the parameters ($t5, $t4, and 0)?

ANSWER:
This looks like it went “t5 += t4”
In other words, at first sight, it appeared to had incremented t5 by t4.

QUESTION:
What is different about how ‘add’ and ‘addi’ work? PROTIP: Look at the code from Part 1 and determine what is similar between instructions that use ‘li’ and instructions that use ‘addi’. Also look at the parameters given to these instructions.

ANSWER:
‘add’ sums two number and stores their values in a new memory. (preservers old values)
‘addi’ sums two numbers but stores the sum in that of an old memory. (increments)

QUESTION:
At step 0x0040003c you will see ‘sll $t6, $t4, 2’. Press F10 and look at the value in $t6. Compare it to the value in $t4 that was used to generate it. What is the mathematical relationship between these two numbers? (Do not worry about the meaning of sll for now.)

ANSWER:
The value in t6 is 0xBA4 which is four times greater than the value at t4 due to the bitwise left shift.

QUESTION:
The command at 0x00400040 is ‘srl $t7, $t5, 2’. Press F10 to run this command, and then convert the values in $t5 and $t7 to binary and write them below.

ANSWER:
Likewise, due to the right bitwise shift by two, the value in t7 is four times less than it was in t5.

QUESTION:
Examine the new register values, and then write what the meanings of ‘sll’ and ‘srl’ are below

ANSWER:
// pseudo command-method signatures
void sll(int valueToStoreAt, int valueToPerformShiftOp, int amountToShifLeft);
void srl(int valueToStoreAt, int valueToPerformShiftOp, int amountToShifRight);

QUESTION:
The value ‘1’ is loaded into register $v0. This indicates that when ‘syscall’ is executed (on the third line) the processor should execute system call #1. On the second line, $a0 is loaded with the value in $t6. This system requires a value in $a0 carry out its function. Write the value of $t6 displayed in the top section of the SPIM window.

ANSWER:
R14 [t6] = ba4

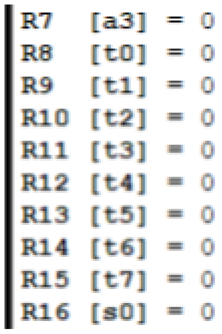
QUESTION:
Now press F10 once more to execute the last line of the code segment above. A window titled ‘Console’ should appear and have the number ‘2980’ displayed at the top. What exactly does system call #1 do?

ANSWER:
It printed the value of t6 to the console.

QUESTION:
Press F10 three times to execute all these instructions. What is the difference between this system call and the previous system call?

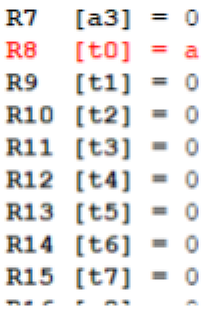
ANSWER:
It printed a string to a console.

Screenshots Next Page:



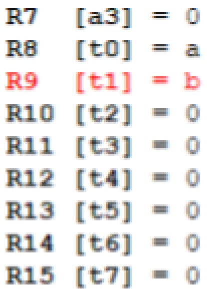
```
R7   [a3] = 0
R8   [t0] = 0
R9   [t1] = 0
R10  [t2] = 0
R11  [t3] = 0
R12  [t4] = 0
R13  [t5] = 0
R14  [t6] = 0
R15  [t7] = 0
R16  [s0] = 0
```

At step 0 all relevant registers are zero



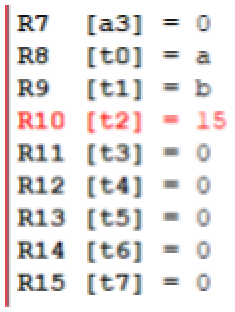
```
R7   [a3] = 0
R8   [t0] = a
R9   [t1] = 0
R10  [t2] = 0
R11  [t3] = 0
R12  [t4] = 0
R13  [t5] = 0
R14  [t6] = 0
R15  [t7] = 0
```

At step 1 the value 10 is put in t0



```
R7   [a3] = 0
R8   [t0] = a
R9   [t1] = b
R10  [t2] = 0
R11  [t3] = 0
R12  [t4] = 0
R13  [t5] = 0
R14  [t6] = 0
R15  [t7] = 0
```

At step 2 eleven is loaded into t1.



```
R7   [a3] = 0
R8   [t0] = a
R9   [t1] = b
R10  [t2] = 15
R11  [t3] = 0
R12  [t4] = 0
R13  [t5] = 0
R14  [t6] = 0
R15  [t7] = 0
```

Lastly the sum of the two are put in t2.