Assignment #1 Key

(max = 80)

Work through the "Notes for Assignment #1" on the course website (under the Course Notes tab). These notes will assist you in the installation of QtSpim on your computer and will give you an opportunity to experiment with QtSpim.

Next, read the first 6 sections of Chapter 1 in the *Computer Organization and Design* text. This reading will give you a glimpse of what we will be looking at during the term.

Afterwards, submit answers for the following problems (where appropriate, show the details of your calculations):

1. When working through the "Notes for Assignment #1", you created a small assembly program called **hello.s**. Remove the period from the ".text" line in the program. Try loading this modified program into QtSpim. (2 points)

   a) What type of error do you get?

      You get a (parser) syntax error.

   b) Does the program load and execute properly if you get rid of the "text" line completely?

      Yes.

2. I assume that you have just executed **hello.s** and are still inside of QtSpim. There are actually several different sections in the QtSpim screen. On the left side you are able to view the contents of the various MIPS integer and floating point registers; we will be learning all about these registers in the coming weeks. Make sure that "Int Regs" is selected. Notice that the PC register contains the value 400038. The General Purpose Registers are numbered starting with 0. Register 1 (also called "at") contains the value 10010000. (8 points)

   a) Register 2 goes by what other name? What value is in register 2?

      (v0) = a

   b) Register 15 goes by what other name? What value is in register 15?

      (t7) = 0

   c) Register 28 goes by what other name? What value is in register 28?

      (gp) = 10008000

d) Register 31 goes by what other name? What value is in register 31?

(ra) = 400018

3. Let’s now look at the right side of the QtSpim screen. We can view our data and our code (“Text”) on this side. Make sure that “Text” is selected. We are able to see the instructions as they are stored in memory. For example, location 00400000 contains the value 8fa40000, which is the instruction “lw $4, 0($29)”. (8 points)

a) What value is stored at location 0040000c? What instruction is this?

00041080 sll $2, $4, 2

b) What value is stored at location 00400034? What instruction is this?

3402000a ori $2, $0, 10

c) What value is stored at location 00400038? What instruction is this?

0000000c syscall

d) What value is stored at location 80000184? What instruction is this?

3c019000 lui $1, -28672

4. Let’s move on to the data portion of the QtSpim screen (still assuming that we just executed **hello.s**). We are able to see the data (and the stack) as it is stored in memory. For example, location 90000030 contains the value 2000205d. (2 points)

a) What value is stored at location 10010000?

6c6c6548

b) What value is stored at location 90000160?

63614d5b

5. The bottom of the QtSpim screen contains some copyright information. (2 points)

a) Who developed the SPIM product (who has the original copyright)?

James R. Larus

b) What year was the original copyright established?

1990

6. I have created another small assembly program called **bio.s** that gives a little bit of information about yours truly. Here is the output that appears in the Console window when I execute the program:

```
My name is Steve Leach
I was born in Key West, Florida
I am a faculty member at FSU's Panama City Campus
I enjoy reading, singing karaoke and playing with my grandchildren
```

   Create a **bio.s** program of your own that tells me a little about yourself. Make sure that your program is documented using a style similar to that used in **hello.s**. Submit a separate file called **bio.s** as well as placing your code in this assignment submission; the Mentor will clarify what I mean by this. (15 points)

   Obviously, answers will vary. Here is my program:

```
# Stephen P. Leach -- 09/30/15
# bio.s -- tell a little bit about myself
# Register use:
#    $vo     syscall parameter and return value
#    $a0     syscall parameter

    .text
    .globl   main
main:
    la       $a0, msg1       # address of line 1 of my bio
    li       $v0, 4          # this is the print_string option
    syscall                  # perform the system call

    la       $a0, msg2       # address of line 2 of my bio
    syscall                  # perform the system call

    la       $a0, msg3       # address of line 3 of my bio
    syscall                  # perform the system call

    la       $a0, msg4       # address of line 4 of my bio
    syscall                  # perform the system call

    li       $v0, 10         # this is the exit option
    syscall                  # perform the system call

# Here is the data for the program
    .data
msg1:    .asciiz  "My name is Steve Leach\n"
msg2:    .asciiz  "I was born in Key West, Florida\n"
msg3:    .asciiz  "I am a faculty member at FSU's Panama City Campus\n"
msg4:    .asciiz  "I enjoy reading, singing karaoke and playing with my grandchildren\n"

# end bio.s
```

7. Do Exercise 1.4 on Page 55 in the textbook. For part (b), give answer to nearest millisecond. (4 points … 2 points each)

a. Frame size = 1280 * 1024 = 1310720 pixels = 1310720 * 3 = 3932160 bytes.
b. Transmission time = (3932160 * 8 bits) / (100 * 10^6 bits/second)
   = .3145728 seconds = 315 ms

8. Do Exercise 1.5 on Page 55 in the textbook. For part (a), simply compute the instructions/second (four significant digits) for each of the three processors (3 points). Part (b) will have six answers: the number of cycles and the number of instructions (four significant digits) for each of the three processors (6 points). For part (c), compute the new execution time and the new CPI for each of the three processors (4 points) and then compute the new required clock rate for each of the three processors, expressing these rates in GHz (four significant digits) (6 points). (19 points total)

   a. P1: (3 * 10^9) / 1.5 = 2.0 * 10^9 instructions/second
      P2: (2.5 * 10^9) / 1.0 = 2.5 * 10^9 instructions/second
      P3: (4 * 10^9) / 2.2 = 1.818 * 10^9 instructions/second
   b. Cycles (P1) = 10 * 3 * 10^9 = 30 * 10^9
      # Instructions (P1) = (30 * 10^9) / 1.5 = 20 * 10^9
      Cycles (P2) = 10 * 2.5 * 10^9 = 25 * 10^9
      # Instructions (P2) = (25 * 10^9) / 1.0 = 25 * 10^9
      Cycles (P3) = 10 * 4 * 10^9 = 40 * 10^9
      # Instructions (P3) = (40 * 10^9) / 2.2 = 18.18 * 10^9
   c. New execution time = 70% * 10 seconds = 7 seconds
      New CPI (P1) = 1.5 * 120% = 1.8
      New CPI (P2) = 1.0 * 120% = 1.2
      New CPI (P3) = 2.2 * 120% = 2.64
      New clock rate (P1) = (20 * 10^9 * 1.8) / 7 = 5.143 * 10^9 = 5.143 GHz
      New clock rate (P2) = (25 * 10^9 * 1.2) / 7 = 4.286 * 10^9 = 4.286 GHz
      New clock rate (P1) = (18.18 * 10^9 * 2.64) / 7 = 6.856 * 10^9 = 6.856 GHz

9. This is an adaptation of Exercise 1.6 on Page 55 in the textbook. We assume that there are two different implementations, P1 and P2, of the same instruction set architecture. The instructions are divided into four classes according their CPIs (classes A, B, C and D). P1 has a clock rate of 2.5 GHz and CPIs of 1, 2, 3 and 3 for the corresponding classes of instructions. Likewise, P2 has a clock rate of 3 GHz and CPIs of 2, 2, 2 and 2.

   We next assume that we have a program that executes a million (10^6) instructions such that 10% of these instructions are of class A, 20% class B, 50% class C and 20% class D. Finally, the question: (8 points total)

   a) For our program, how many instructions of each class are executed? (2 points)

      Class A: 10% * 10^6 = 10^5 instructions
      Class B: 20% * 10^6 = 2 * 10^5 instructions
      Class C: 50% * 10^6 = 5 * 10^5 instructions
      Class D: 20% * 10^6 = 2 * 10^5 instructions

b) Compute the execution time of the program using P1 and using P2. Express the answer to the nearest microsecond. (6 points … 3 points each)

Execution time (P1) = $(10^5*1 + 2*10^5*2 + 5*10^5*3 + 2*10^5*3) / (2.5*10^9)$
$= 10.4 * 10^{-4}$ seconds $= 1040 * 10^{-6}$ seconds = 1040 µs
Execution time (P2) = $(10^5*2 + 2*10^5*2 + 5*10^5*2 + 2*10^5*2) / (3*10^9)$
$= 6.667 * 10^{-4}$ seconds $= 666.7 * 10^{-6}$ seconds = 667 µs

10. Do Exercise 1.10.1 on Page 57 in the textbook. As part of this problem, you will want to derive a formula for die area (see page 28 of the textbook; note that this will be an approximation). For both wafers, give your calculated value (use 3.14 for π) of die area (in cm^2, two decimal places) and the value of yield to four decimal places. (8 points … 2 points each)

Dies per wafer ≈ (Wafer area)/(Die area) => Die area ≈ (Wafer area)/(Dies per wafer)
Die area (15 cm wafer) = (3.14 * 7.5 * 7.5) / 84 = 2.10 cm^2
Yield (15 cm wafer) = $1/(1+(0.020*2.10/2))^2 = 0.9593$
Die area (20 cm wafer) = (3.14 * 10 * 10) / 100 = 3.14 cm^2
Yield (20 cm wafer) = $1/(1+(0.031*3.14/2))^2 = 0.9093$

11. Do Exercise 1.10.2 on Page 57 in the textbook. The problem does not specify the unit of cost (it is simply 12 and 15). Therefore, your answer will not have a specific unit either. Give answer to four decimal places. (4 points … 2 points each)

Cost per Die (15 cm wafer) = 12 / (84 * 0.9593) = 0.1489
Cost per Die (20 cm wafer) = 15 / (100 * 0.9093) = 0.1650

Your assignment is due by 11:59 PM (Eastern Time) on the assignment due date (consult Course Calendar on course website).