EE 352 Homework 3
Spring 2010 Nazarian

Name: _____ **Score:_____**

Assigned: Friday, February 19

Due: Thursday, March 4 at 9:30am (in class)

Exercise 2.18 from the textbook

1) For these problems, the following table holds MIPS assembly code fragments. You will be asked to evaluate each of the code fragments, familiarizing you with the different MIPS branch instructions.

A		addi $t1, $0, 100
	LOOP:	lw $s1, 0($s0)
		add $s2, $s2, $s1
		addi $s0, $s0, 4
		subi $t1, $t1, 1
		bne $t1, $0, LOOP
B		addi $t1, $0, 400
	LOOP:	lw $s1, 0($s0)
		add $s2, $s2, $s1
		lw $s1, 4($s0)
		add $s2, $s2, $s1
		addi $s0, $s0, 8
		bne $t1, $0, LOOP

(2.18.4) What is the total number of MIPS instructions executed?
(2.18.5) Translate the loops above into C. Assume that the C-level integer i is held in register $t1, $s2 holds the C-level integer called result, and $s0 holds the base address of the integer MemArray.
(2.18.6) Rewrite the loop in MIPS assembly to reduce the number of MIPS instructions executed.
Note: The loop in parts a and b implement the same high-level algorithm/code. Thus, for parts 2.18.5 and 2.18.6 you need only submit one answer (i.e. for 2.18.5 translate the loops to the equivalent C code and then for 2.18.6 rewrite that algorithm to minimize the number of executed instructions)

Subroutines and Assembly

2) In addition to passing arguments in registers or by using the stack, another possible method (that no one would ever use) is to provide space for the arguments in the code itself (though you should note that this code will NOT assemble using MARS because it does not allow data directives in the text section). However please study the code below to understand how this method works and answer the given questions. Show any work on an extra page and submit that with your homework. **SHOW ALL VALUES in HEX.**

Assume that the .data section starts at 0x10010000.

a) **Before execution of** "**jal AVG**", what are the contents of $t0, $t1, and $t2.
b) **Before execution of "sra $v0,$v0,1"**, what are the contents of $v0, $t3, & $ra?
c) **After program execution**, what are the contents of $ra and the word at address RES.

```
                .data
          VALS: .half  0xbead, 0xface
          RES:  .space  4
                .text
0x400000        la          $t0,VALS
0x400008        lh          $t1,($t0)
0x40000c        lhu         $t2,2($t0)
0x400010        sll         $t2,$t2,16
0x400014        or          $t2,$t1,$t2
0x400018        jal         AVG
0x40001c        .word       -2
0x400020        .word       -6
0x400024        la          $t4,RES
0x40002c        sw          $v0,($t4)
0x400030        li          $v0,10
0x400038        syscall

          AVG:  lw          $v0,($ra)
                lw          $t3,4($ra)
                add         $v0,$v0,$t3
                sra         $v0,$v0,1
                addi        $ra,$ra,8
                jr          $ra
```

a) $t0 = 0x_____
   $t1 = 0x_____
   $t2 = 0x_____

b) $v0 = 0x_____
   $t3 = 0x_____
   $ra = 0x_____

c) $ra = 0x_____
   M[RES] = 0x_____

3) Consider the following assembly code.
a) Show the word content (in hex) of registers $t0,$t1,$sp and the PC:
I. Just after execution of the `jal` instruction at location L1
II. Just after the first execution of the `jal` instruction at L2
III. Just after the first `jr $ra` instruction execution
IV. After execution of the `li $v0,10` instruction
b) Show the contents of the stack at its fullest.
c) What arithmetic function is this code performing?

```
                    .text
0x400000  BEGIN:  addi   $t0,$zero,-1
0x400004          li     $t1,16
0x40000c  L1:     jal    SUB1
0x400010          li     $v0,10
0x400018          syscall
                  ...
0x4c0000  SUB1:   beq    $t1,$0,EXIT
0x4c0004          addi   $sp,$sp,-4
0x4c0008          sw     $ra,0($sp)
0x4c000c          srl    $t1,$t1,1
0x4c0010          addi   $t0,$t0,1
0x4c0014  L2:     jal    SUB1
0x4c0018          lw     $ra,0($sp)
0x4c001c          addi   $sp,$sp,4
0x4c0020  EXIT:   jr     $ra
```

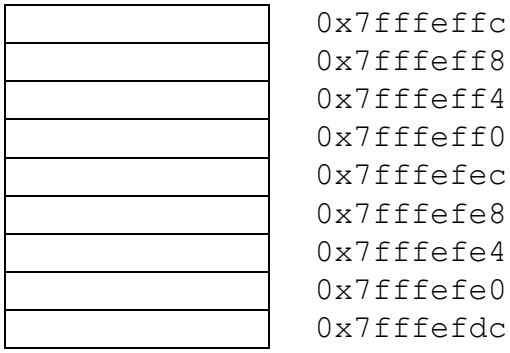
Instructions: Enter the hex values for registers and memory (include the 0x). For part c., describe the operation.

a)

	$t0	**$t1**	**$sp**	**PC**
I)				
II)				
III)				
IV)				

b)

Initially
$sp = 0x7fffeffc -->



c) What function or operation is this code performing (Hint: try $t1 = 32 or 8)?

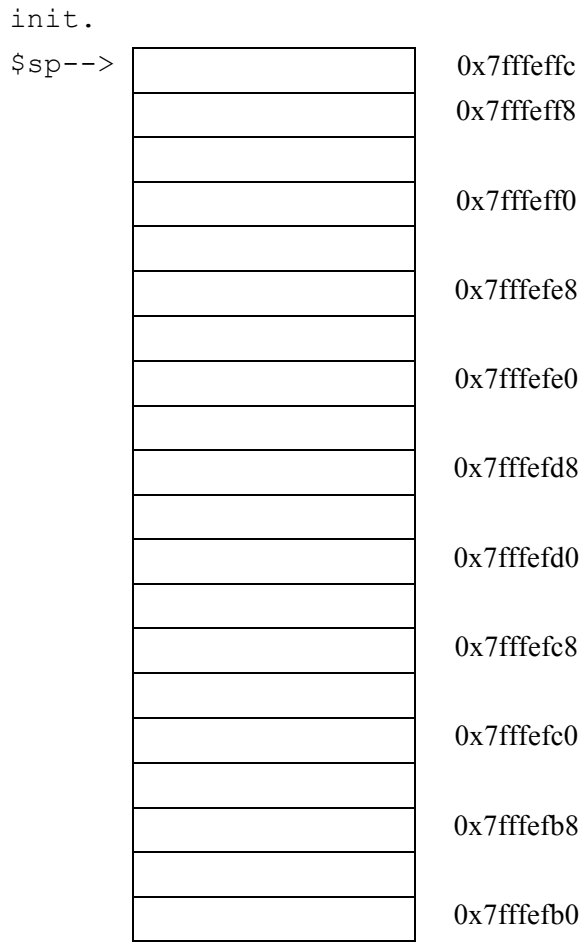
Stack Frames

4) Examine the C-style subroutines and the actual "gcc" MIPS assembly output below. Assume that we start execution at the beginning of the "caller" routine. Further, assume that initially, $sp = $fp = 0x7fffeffc, $ra=0x00400018, $gp=0x10018000. Further assume that the argument i passed into the "caller" routine equals 0x2. RA for the call to AVG is 0x0040ce18. Follow the assembly code and fill in the stack data values (in hex) that this trace would create. Because of how "gcc" works, the stack frames are a bit more bloated than they need to be and many locations will go unused. Fill unused locations with 0's. Try to understand the correspondence of variables to their particular locations in the stack frame.

```
void caller(int i, short j)
{
  unsigned int a=5;
  a = avg(a,3);
}
unsigned int avg(unsigned int x, unsigned int y)
{
  int temp;
  temp = (x+y)/2;
  return temp;
}
```

```
caller:
        subu    $sp,$sp,48
        sw      $31,40($sp)
        sw      $fp,36($sp)
        sw      $28,32($sp)
        move    $fp,$sp
        sw      $4,48($fp)
        li      $2,5                    # 0x5
        sw      $2,24($fp)
        lw      $4,24($fp)
        li      $5,3                    # 0x3
        la      $25,avg
        jalr    $31,$25
        sw      $2,24($fp)
        move    $sp,$fp
        lw      $31,40($sp)
        lw      $fp,36($sp)
        addu    $sp,$sp,48
        jr      $31
avg:    subu    $sp,$sp,24
        sw      $fp,20($sp)
        sw      $28,16($sp)
        move    $fp,$sp
        sw      $4,24($fp)
        sw      $5,28($fp)
        lw      $3,24($fp)
        lw      $2,28($fp)
        addu    $2,$3,$2
        srl     $2,$2,1
        sw      $2,8($fp)
        lw      $2,8($fp)
        move    $sp,$fp
        lw      $fp,20($sp)
        addu    $sp,$sp,24
        jr      $31
```

Instructions: Enter the hex values for each memory location (include the 0x). Fill unused locations with 0s.



	0x7fffeffc
	0x7fffeff8
	0x7fffeff0
	0x7fffefe8
	0x7fffefe0
	0x7fffefd8
	0x7fffefd0
	0x7fffefc8
	0x7fffefc0
	0x7fffefb8
	0x7fffefb0

Exceptions

5) Refer to pages 19 and 20 of the "exceptions" slide units. Rewrite the program for invalid address during store instruction

a) Similarly to page 19 setup, write a sample user code that has a sw, or sh instruction in it. Then assume in invalid address exception occurs, therefore the processor needs to call the corresponding exception handler similarly to what is shown on that page for the case of load instruction.

b) Similarly to page 19 code on the right side of the page you will write the code that detects the cause as "invalid address during a store instruction"

Note: You will follow what you see on page 19, this means you do not need to write the code for store exception handler.

Performance and Amdahl's Law

6) Exercises 1.3, 1.5, 1.6 and 1.15 from the textbook

a) Exercises 1.3.1-1.3.3 in CO&D, 4th Ed. page 59:

Consider three different processors P1, P2, and P3 executing the same instruction set with the clock rates and CPIs given in the following table:

Processo	Clock Rate	CPI	Processor
P1	2 GHz	1.5	P1
P2	1.5GHz	1.0	P2
P3	3 GHz	2.5	P3

1.3.1 Which processor has the highest performance?

1.3.2 If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.

1.3.3 We are trying to reduce the time by 30% but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reduction?

b) Exercises 1.5 (1.5.1-1.5.6) in CO&D, 4th Ed. page 60-61. For problems 1.5.2-1.5.3 when it says that class A (or E) occurs twice as often as each of the others, let us assume that means that for every 6 instructions, on average two would be class A (or E) and one instruction from every other class:

Consider two different implementations, P1 and P2, of the same instruction set. There are 5 classes of instructions, A, B, C, D, and E, in the instruction set. The clock rate and CPI of each class is given below.

		Clock Rate	CPI Class A	CPI Class B	CPI Class C	CPI Class D	CPI Class E
a.	P1	1.0GHz	1	2	3	4	3
	P2	1.5GHz	2	2	2	4	4
b.	P1	1.GHz	1	1	2	3	2
	P2	1.5GHz	1	2	3	4	3

1.5.1 Assume that peak performance is defined as the fastest rate that a computer can execute any instruction sequence. What are the peak performances of P1 and P2 expressed in instructions per second?

1.5.2 If the number of instructions executed in a certain program is divided equally among the classes of instructions except for class A, which occurs twice as often as each of the others. Which computer is faster? How much faster is it?

1.5.3 If the number of instructions executed in a certain program is divided equally among the classes of instructions except for class E, which occurs twice as often as each of the others? Which computer is faster? How much faster is it?

The table below shows instruction-type breakdown for different programs. Using this data you will be exploring the performance tradeoffs with different changes made to a MIPS processor.

		# Instructions			
		Compute	Load	Store	Branch
a.	Program 1	1000	400	100	50
b.	Program 2	1500	300	100	100

1.5.4 Assuming that computes take 1 cycle, loads and store instructions take 10 cycles, and branches take 3 cycles, find the execution time of each program on a 3GHz MIPs processor.

1.5.5 Assuming that computes take 1 cycle, loads and store instructions take 2 cycles, and branches take 3 cycles, find the execution time of each program on a 3 GHz MIPs processor.

1.5.6 Assuming that computes take 1 cycle, loads and store instructions take 2 cycles, and branches take 3 cycles, what is the speed-up of a program if the number of compute instruction can be reduced by one-half?

c) Exercises 1.6.1 - 1.6.3 in CO&D, 4th Ed. page 61-62:

Compilers can have a profound impact on the performance of an application on a given processor. This problem will explore the impact compilers have on execution time.

	Compiler A		Compiler B	
	# Instruction	Execution time	# Instructions	Execution time
a.	1.00E+09	1s	1.20E+09	1.4s
b.	1.00E+09	0.8s	1.20E+09	0.7s

1.6.1 For the same program, two different compilers are used. The table above shows the execution time of the two different compiled programs. Find the average CPI for each program given that the processor has a clock cycle time of 1ns.

1.6.2 Assume the average CPIs found in 1.6.1, but that the compiled programs run on two different processors. If the execution times on the two processors are the same, how much faster is the clock of the processor running compiler A's code versus the clock of the processor running compiler B's code?

1.6.3 A new compiler is developed that uses only 600 million instructions and has an average CPI of 1.1. What is the speed-up of using this new compiler versus using Compiler A or B on the original processor of 1.6.1?

d) Exercises 1.15.1 - 1.15.3 in CO&D, 4th Ed. page 71.

Consider a computer running programs with CPU times shows in the following table:

	FP Instr.	INT Instr.	L/S Instr.	Branch Instr.	Total time
a.	35s	85s	50s	30s	200s
b.	50s	80s	50s	30s	210s

1.15.1 By how much is the total time reduced if the time for FP operations is reduced by 20%?

1.15.2 By how much is the time for INT operations reduced if the total time is reduced by 20%?

1.15.3 Can the total time be reduced by 20% by reducing only the time for branch instructions?

Notes: For 1.15.1 enter the new total time. For 1.15.2 assume that only the INT operations are improved and all others stay the same.