NAME

First Last

Student ID#

**STONY BROOK UNIVERSITY**

**COMPUTER SCIENCE DEPARTMENT**

**PRACTICE MIDTERM**

CSE 220

Fall Semester 2014

Ignore the points on each question.

This practice ”exam” includes extra questions to help you prepare.

All questions are from old exams.

[1 pt each] True/False & Multiple Choice. Write in the correct answer.

|  |  |
| --- | --- |
| **T / F** | Register $0 of MIPS always stores 0. |
| **T / F** | The Program Counter (PC) is a MIPS ‘general purpose’ registers. |
| **T / F** | Signed Magnitude number representation has 2 representations for 0, all zeros and all ones. |
| **T / F** | 2’s complement number representation has an equal number of positive & negative numbers. |
| **T / F** | Excess-1023 is used for the exponent in double precision IEEE 754 floating point notation. |
| **T / F** | A byte holds 4 ASCII characters. |

|  |
| --- |
| 1 |

Register $sp

1. always points to the bottom of the stack
2. always points to the top of the stack
3. is reserved for the O/S
4. is reserved for the assembler

Which of the following is NOT TRUE for MIPS functions

1. non-floating point parameters are passed using registers $a0-$a4
2. all return values are placed in $v1
3. instruction *jal* is used to call the function
4. a value can not be saved to the $zero register

The I instruction format does not contain which of the following fields:

1. Target register
2. Source register
3. Function
4. Opcode
5. Immediate

The branch on equal (’beq’) instruction uses which format:

1. Register ( R )
2. Immediate ( I )
3. Jump ( J )
4. None of the above

A calling function (a function which calls another function) is responsible for

1. placing parameters on the stack
2. saving the $s0-$s7 registers
3. saving the $t0-$t7 registers
4. moving the $fp

**2**

Which of the following is NOT a MIPS instruction (including pseudo instructions)

1. shift right arithmetic *sra*
2. shift right logical *srl*
3. shift left arithmetic *sla*
4. shift left logical *sll*
5. None of the above

Consider a 16 bit register, $r which contains a 16-bit binary value 1001 1000 1111 0011. What are the contents of $r after the following instructions are executed?

sra $r, $r, 3 ori $r, $r, 127

1. 1100 1100 1111 1011
2. 0100 1100 0111 1111
3. 0000 0000 0111 1000
4. 1111 0011 0111 1111

A 6 digit number in base 16 has digits in base 8.

(a) 5 **(b) 8** (c) 9 (d) 11 (e) 12

When we call function A from function B, the return address of function A is stored in what register of MIPS?

(a) $v0 (b) $a0 **(c) $ra** (d) $sp

Assume that registers are 8 bit wide. Register $t0 contains 0000 1100 and $t1 contains 0000 1001. What does $t2 contain after executing instruction xor $t2, $t0, $t1?

(a) 0000 1101 (b) 0000 0101 (c) 1111 0010 (d) 0000 1000

Consider an 8-bit (including sign bit) 2’s complement number. Which 8-bit number does not have a corresponding opposite value (ie. x = - x)?

(a) 256 (b) -128 (c) -256 (d) -127

Convert -3410 to an 8-bit (including sign) 1’s complement number.

(a) 1101 1101 (b) 1011 1101 (c) 1101 1010 (d) 1101 1101

In IEEE double precision format the exponent is bits.

(a) 1 (b) 4 (c) 8 (d) 11 (e) 32

Which of the following special values ARE NOT represented in the IEEE format.

(a) ±0 (b) ±NaN (c) ±Inf (d) ±Null

2 (8 pts) Number Representation

Some number formats have positive and negative representations for zero. Write all forms of zero for each of the following number representations.

1. Sign Magnitude (using 6- bits ) 000000, 100000
2. 1’s complement (using 6- bits )000000,111111
3. 2’s complement (using 6- bits )000000
4. IEEE single precision floating point (state value of each field, do not write out all bits)
5. **(v)** Consider the IEEE double precision floating point number representation.
   1. How many bits are used for representing the fraction/mantissa? 23
   2. Normalized numbers use excess- system to represent the exponent. 1023
   3. For positive or negative zero, the exponent value in binary is 0

3 Number conversion/IEEE Floating Point

1. (2 pts) Convert -9FA.4C16 to base 8. Show all steps.
2. (2 pts) Convert -9FA.4C16to normalized binary. Present the result in binary product (ie. sign\*signpenisificand\*2*exponent*).
3. (2 pts) State the sign bit, exponent, and fraction in IEEE 754 single precision binary format for the normalized number in ( ii ).

4 Number Representation

1. (4 pts) Convert 19.710 to base 3. Show all steps to get full/partial credit. You must have at least two digits after the decimal point.
2. (4 pts) What is the largest 3-digit number in base 7? (Answer in base 7). Convert the base 7 number to decimal.

5 Number Representation

Consider a decimal number X = -17. We want to store it in a 6-bit long word that includes the sign bit. What are all five bits for the following representations of -17 ?

1. (2 pts)Signed Magnitude:
2. (2 pts) 1’s complement: **(iii)** (2 pts)2’s complement:

Consider numbers A = 010112 and B = 111012 in 2’s complement form. Numbers are five bits long including the sign bit.

|  |
| --- |
| 6 |

1. (2 pts) What is the decimal value of B.
2. (2 pts) Perform *A* − *B*. Show all steps and explain your answer.
3. (3 pts) Did an overflow/underflow occur? If No, for what values of B would it occur. If Yes, for what values of B would it not occur.

7 [8 pts] Pseudo-instructions

1. [2 pts] What is a pseudo-instruction?
2. [2 pts] The instruction *la* $*t*0*,A* is a pseudo instruction. A is a label declared in the .data section. Briefly explain how you can tell it is not an actual machine instruction.
3. [4 pts] MIPS has a pseudo-instruction *neg*. It obtains the 2’s complement of an integer. For example, *neg* $*t*0*,*$*t*1 calculates the 2’s complement of $t1 and stores it in $t0. How would the assembler implement the *neg* instruction (ie. write the equivalent MIPS instructions)?
4. [10 pts] Translate the following code into MIPS assembly:

A[i] = (2\*B[i-1]) + (B[i]-3) + B[i+4]

Assume that A is a word array and B is a byte array. Both arrays are stored in memory. Register $s1

contains the initial address for array A; register $s2 contains the initial address for array B; and register $s3 contains the value of i. Note, this is not a loop, only a statement. A

ssume that all calculated i values are within array bounds. No error checking is necessary.

1. [10 pts] Consider the following .data segment.

.data

X: .word 300

Y: .byte 500

A: .space 100

.align 2

B: .word 1 ,2,3,4, 5

Str: .asciiz "I love MIPS!" End: .ascii "Goodbye!"

1. [2 pts] One of the variable declarations has an error. Name the variable and explain the error.
2. [2 pts] What does the *.align* directive do? Why?
3. [2 pts] What is the difference between variables Str and End?
4. [2 pts] Assume the error has been fixed and the .data section is allocated in the same order. What is the size of the data segment (in bytes)?
5. [2 pts] Assume the data segment starts at 0x10010000, what is the base address of A?

10 [20 pts] The Ackermann function is the simplest example of a well defined total function which is computable but not primitive recursive. Write MIPS assembly code for the following Java program fragment, which computes the Ackermann function. You must follow all MIPS function call standards. You may use pseudo instructions, but points will be taken off for inefficient code.

(Hint: Don’t forget to use the stack.)

int Ackermann(int x, int y)

{ int result; if(x == 0) result = y + 1 ;

else if(y == 0) result = Ackerman(x-1,1)

else result = Ackerman(x-1, Ackerman(x, y-1));

return result;

}

11 [20 pts] The following code loads the address of the x and y arrays, calls the Ackermann function for the values, and prints the result to the screen.

.text

|  |  |  |  |
| --- | --- | --- | --- |
| .globl main | |  |  |
| main: | la $s0, x la $s1, y add $t3, $0, $0 |
| loop: | bge $t3, 3, end sll $t4, $t3, 2 add $t0, $s0, $t4 add $t1, $s1, $t4 lw $a0, ( $t 0) lw $a1, ( $t 1) |  |  |
|  | ....  jal Ackermann | # <---- | Part ( a ) |
|  | ....  add $a0, $0, $v0 li $v0, 1  syscall addi $t3, $t3, 1 j loop | # <---- | Part ( b ) |
| end: .data | .... syscall | # <---- Part ( c ) | |
| x: | .word 1 ,2,3, 3 |  | |
| y: | .word 0 ,2,4, 5 |  | |

**Output:**

2

7

125

253

1. [9 pts] What should happen before the function call? Why? Implement the necessary instructions.
2. [9 pts] What should happen after the function call? Implement the necessary instructions.
3. [2 pts] What syscall should be used to exit the program? What are the parameters, if any

Extra page provided as additional work space.