

Date:_____

Quiz for Chapter 2 Instructions: Language of the Computer

Not all questions are of equal difficulty. Please review the entire quiz first and then budget your time carefully.

Name:_____

Course:_____

1. [5 points] Prior to the early 1980s, machines were built with more and more complex instruction set. The MIPS is a RISC machine. Why has there been a move to RISC machines away from complex instruction machines?

Name: _____

2. [5 points] Write the following sequence of code into MIPS assembler:

$x = x + y + z - q;$

Assume that x , y , z , q are stored in registers $\$s1 - \$s4$.

Name: _____

3. [10 points] In MIPS assembly, write an assembly language version of the following C code segment:

```
int A[100], B[100];
for (i=1; i < 100; i++) {
    A[i] = A[i-1] + B[i];
}
```

At the beginning of this code segment, the only values in registers are the base address of arrays A and B in registers \$a0 and \$a1. Avoid the use of multiplication instructions—they are unnecessary.

Name: _____

4. [6 points] Some machines have a special flag register which contains status bits. These bits often include the *carry* and *overflow* bits. Describe the difference between the functionality of these two bits and give an example of an arithmetic operation that would lead to them being set to different values.

Name: _____

5. [6 points] The MIPS instruction set includes several shift instructions. They include logical-shift-left, logical-shift-right, and arithmetic-shift-right. Other architectures only provide an arithmetic-shift-right instruction.

a) Why doesn't MIPS offer an "arithmetic-shift-left" opcode?

b) How would you implement in the assembler a logical-shift-left (LSL) pseudo-operation for a machine that didn't have this particular instruction? Be sure your LSL instruction can shift up to W-bits where W is the machine word size in bits.

Name: _____

6. [6 points] Consider the following assembly code for parts 1 and 2.

```
r1 = 99
```

Loop:

```
    r1 = r1 - 1  
    branch r1 > 0, Loop  
halt
```

(a) During the execution of the above code, how many dynamic instructions are executed?

(b) Assuming a standard unicycle machine running at 100 KHz, how long will the above code take to complete?

Name: _____

7. [15 points] Convert the C function below to MIPS assembly language. Make sure that your assembly language code could be called from a standard C program (that is to say, make sure you follow the MIPS calling conventions).

```
unsigned int sum(unsigned int n)
{
    if (n == 0) return 0;
    else return n + sum(n-1);
}
```

This machine has no delay slots. The stack grows downward (toward lower memory addresses). The following registers are used in the calling convention:

Register Name	Register Number	Usage
\$zero	0	Constant 0
\$at	1	Reserved for assembler
\$v0, \$v1	2, 3	Function return values
\$a0 - \$a3	4 - 7	Function argument values
\$t0 - \$t7	8 - 15	Temporary (caller saved)
\$s0 - \$s7	16 - 23	Temporary (callee saved)
\$t8, \$t9	24, 25	Temporary (caller saved)
\$k0, \$k1	26, 27	Reserved for OS Kernel
\$gp	28	Pointer to Global Area
\$sp	29	Stack Pointer
\$fp	30	Frame Pointer
\$ra	31	Return Address

Name: _____

8. [5 points] In the snippet of MIPS assembler code below, how many times is instruction memory accessed? How many times is data memory accessed? (Count only accesses to memory, not registers.)

```
lw $v1, 0($a0)
addi $v0, $v0, 1
sw $v1, 0($a1)
addi $a0, $a0, 1
```


Name: _____

9. [6 points] Use the register and memory values in the table below for the next questions. Assume a 32-bit machine. Assume each of the following questions starts from the table values; that is, DO NOT use value changes from one question as propagating into future parts of the question.

Register	Value	Memory Location	Value
R1	12	12	16
R2	16	16	20
R3	20	20	24
R4	24	24	28

a) Give the values of R1, R2, and R3 after this instruction: `add R3, R2, R1`

b) What values will be in R1 and R3 after this instruction is executed: `load R3, 12(R1)`

c) What values will be in the registers after this instruction is executed: `addi R2, R3, #16`

10. [20 points] Loop Unrolling and Fibonacci: Consider the following pseudo-C code to compute the fifth Fibonacci number ($F(5)$).

```

1 int a,b,i,t;
2 a=b=1; /* Set a and b to F(2) and F(1) respectively */
3 for(i=0;i<2;i++)
4 {
5 t=a; /* save F(n-1) to a temporary location */
6 a+=b; /* F(n) = F(n-1) + F(n-2) */
7 b=t; /* set b to F(n-1) */
8 }

```

One observation that a compiler might make is that the loop construction is somewhat unnecessary. Since the range of the loop indices is fixed, one can unroll the loop by simply writing three iterations of the loop one after the other without the intervening increment/comparison on i . For example, the above could be written as:

```

1 int a,b,t;
2 a=b=1;
3 t=a;
4 a+=b;
5 b=t;
6 t=a;
7 a+=b;
8 b=t;

```

(a) Convert the pseudo-C code for both of the snippets above into reasonably efficient MIPS code. Represent each variable of the pseudo-C program with a register. Try to follow the pseudo-C code as closely as possible (i.e. the first snippet should have a loop in it, while the second should not).

(b) Now suppose that instead of the fifth Fibonacci number we decided to compute the 20th. How many static instructions would there be in the first version and how many would there be in the unrolled version? What about dynamic instructions? You do not need to write out the assembly for this part.

Name: _____

11. [10 points] In MIPS assembly, write an assembly language version of the following C code segment:

```
for (i = 0; i < 98; i++) {  
    C[i] = A[i + 1] - A[i] * B[i + 2]  
}
```

Arrays A, B and C start at memory location *A000hex*, *B000hex* and *C000hex* respectively. Try to reduce the total number of instructions and the number of expensive instructions such as multiplies.

Name: _____

12. [6 points] Suppose that a new MIPS instruction, called `bcp`, was designed to copy a block of words from one address to another. Assume that this instruction requires that the starting address of the source block be in register `$t1` and that the destination address be in `$t2`. The instruction also requires that the number of words to copy be in `$t3` (which is > 0). Furthermore, assume that the values of these registers as well as register `$t4` can be destroyed in executing this instruction (so that the registers can be used as temporaries to execute the instruction).

Do the following: Write the MIPS assembly code to implement a block copy without this instruction. Write the MIPS assembly code to implement a block copy with this instruction. Estimate the total cycles necessary for each realization to copy 100-words on the multicycle machine.