Computer Science 605.611

Problem Set 4
(Please do not use outside resources such as simulators, disassemblers or web sites to complete this assignment.)

1. (4) Consider the following statement: “Every MIPS assembly language statement is translated by the assembler into a single 32-bit machine code instruction.” Explain why the statement is true or is false.

This statement is false. Many MIPS instructions are pseudo-instructions, meaning that they are shorthand ways of writing multiple consecutive MIPS commands. move, li, mul, blt, and bgt are all examples of MIPS pseudo-instructions, which when handled by the assembler can be made into multiple 32-bit machine code instructions.

2. a) (7) Prior to executing the MIPS machine instruction 0x10A1FFFE (which resides at address 0xB0000000), CPU registers $1 through $31 all contain the pattern 0xFFFFFFFA. What is the hex pattern in the PC (program counter) and any modified CPU general purpose registers ($0 through $31) immediately after this instruction is executed?

0x10A1FFFE -> 00010000010100011111111111111110 -> beq $a1, $at, 0xFFFE
No registers modified
The Program counter = 0xB0000000 + 0xFFFE = 0xB000FFFE

b) (7) Prior to executing the MIPS machine instruction 0x04A1FFFE (which resides at address 0xB0000000), registers $1 through $31 all contain the pattern 0xFFFFFFFA. What is the hex pattern in the PC (program counter) and any modified CPU general purpose registers ($0 through $31) immediately after this instruction is executed?

0x04A1FFFE -> 00000100101000011111111111111110 -> bgez $a1, 0xFFFE
No registers modified
The program counter = 0xB0000000 + 0xFFFE = 0xB000FFFE

3. (4) For our MIPS 9-instruction core subset (add, sub, and, or, slt, lw, sw, beq & jump) which bits within the machine instruction does the control unit examine to decide whether to set the RegDst control bit to 0 or to 1?

The control unit would read the first 6 bits of the MIPS code and make a determination whether the instruction is R, I, or J type. If the function is R-type, RegDst is set to 1. For lw, RegDst is set to 0, and for sw and beq the RegDst bit “does not care”

4. (4) List all assembly instructions within our 9-instruction MIPS core subset for which the control unit would set the MemtoReg control bit to 1 when it processes the corresponding machine instruction.

MemtoReg is asserted when write date input comes from the data memory, and de-asserted when write date input comes from the ALU. Therefore the MIPS instructions where write data input comes from data memory would have the MemtoReg bit set to 1. This would be the case for the lw instruction only. This can be the case for sw and beq, since these instructions "do not care" what the value of the MemtoReg bit is. The remaining 6 R-type instructions would have MemtoReg set to 0.

5. (5) Suppose the instruction jr $t0 resides at memory address 0x04000000. What is the highest valid instruction address (expressed in hex) to which this instruction can transfer control?

The jump register instruction multiplies the rightmost 26 bits in the machine instruction by 4 and prepends the upper 4 bits of the PC to the resulting 28 bits to generate the 32-bit jump address. Hence the highest valid instruction address to which the jr instruction can transfer control is 0x4FFFFFFC.

6. (5) List all assembly instructions within our MIPS 9-instruction core subset whose machine instruction format is the I-type format.

Instruction	Type
lw	I-type
sw	I-type
beq	I-type
add	R-type
sub	R-type
and	R-type
or	R-type
slt	R-type
j	J-type

lw, sw are both I-type

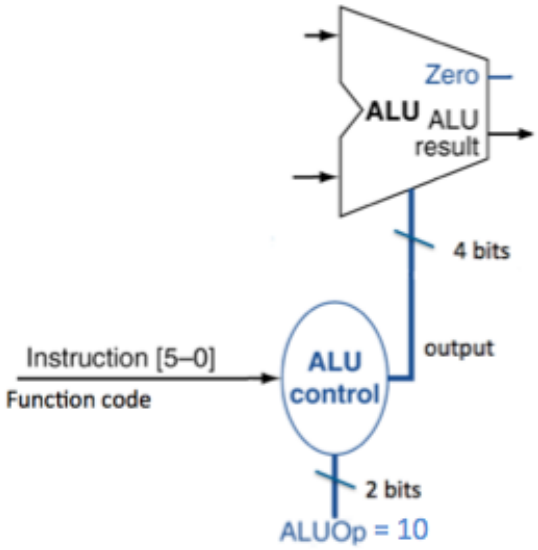
7. (4) The instructions: addi, ori, and lw all use the register identified by the rt field as the result register. Why can't these instructions use the register identified by the rd field as the result register?

Using the rt field as the result register is a convention among R-type instructions. Due to the fact that I-type instructions have no rd field, it is consistent to have both the R-type instructions and I-type instructions reference the same register to store

their results, which makes the rt field a good candidate to store the result of the instruction.

8. (6) The diagram below shows how the ALU control unit directs the ALU during the execution of the instructions within our MIPS core subset. Assuming that the ALUOp bits = 10, complete the table below to show the 4-bit output from the ALU control unit for each of the listed 6-bit function code inputs described in module 4:

Function code	4-bit ALU control output
101010	0111
100100	0000
100000	0010
100101	0001
100010	0110



9. a) (4) Why does the single-cycle datapath require separate instruction and data memories?

In single cycle datapath the memory only has one port, so to run an instruction in a single clock cycle, we need to have separate instruction and data memories.

b) (4) Instead of using a single control bit Memctl to indicate whether to perform a memory read (Memctl =0) or a memory write (Memctl=1) with the single-cycle datapath, why are separate control bits (MemRead and MemWrite) used?

Using a single bit to specify memRead and memWrite would only work if all of your instructions accessed memory (such as lw and sw). Some instructions do not do either (j, beq), therefore the added fidelity of having two bits allows you to specify if you are reading/writing memory, and if you are accessing memory at all.

9. (5) Assume that $6 contains a two's complement integer that is greater than the two's complement integer in $5. After the instruction sub $4,$5,$6 is executed, what is the value of the MSB (bit31) in register $4 if an overflow occurs?

For overflow to occur, the value in $6 must be positive and the value in $5 must be negative. Subtracting a negative number from a positive number will yield a positive result, so a negative value in $4 would indicate overflow. The MSB of 1 would indicate a negative value, so the value of the MSB should be 1.

10. (5) List all instructions within our MIPS core instruction subset (add, sub, and, or, slt, lw, sw, beq & jump) that do not require the output of the sign extension unit.

Sign extension refers to copying the most significant bits to the upper 16 bits in order to keep the same sign information when converting 16 bit numbers to 32 bit numbers. Instructions which do NOT require the output of the sign extension unit include: add, sub, and, or, slt, beq and jump.

11. (10) Assume that the zero flag and registers $0 through $31 all contain 0 prior to executing the machine instruction 0x0C3A7E24 which resides at memory address 0x400B00C8. What value (expressed in hex) will be in any CPU registers that are modified and what value will be in the program counter (PC) register when the instruction completes execution?

0x0C3A7E24 -> 000011 00001110100111111000100100
= jal, 00 0011 1010 0111 1110 0010 0100
$31 is set to 0x400B00CC (=0x400B00C8 + 4)
Program counter = 0000 1110 1001 1111 1000 1001 0000 = 0x0E9F890

12. Show the 8-digit hex representation of the 32-bit machine instruction for each of the following true-op assembly language instructions:
(3) add $4,$5,$6 machine code = 0x00a62020
(3) or $4,$5,$6 machine code = 0x00a62025

13. Consider the MIPS assembly language instruction ori $6,$4,8 .
a) (3) What is the operator mnemonic for this instruction?

ori is the operator mnemonic for the instruction "Bitwise or immediate" which ors a register and an immediate value and stores the result in a register.

b) (3) What is the opcode for this instruction?

001101, or 13

14. Suppose that registers $12 and $11 both contain the two's complement representation of the negative decimal integer -2147483648.

a) (3) What decimal value is represented by the result produced by the instruction: addiu $5,$12,-2 ?

2's compliment -2147483648 = 10000000000000000000000000000000 = 0x80000000
addiu $5,$12,-2 -> $5 = 0x7FFFFFFE = 2147483646
Decimal value = 2147483646

b) (3) What decimal value is represented by the result produced by the instruction: andi $5,$11,-2 ? Decimal value = -2147483648

This is because andi ands a register and an immediate value and stores the result in the register. The only bit that produces a valid AND with 0x80000000 is the first bit, so the result is the same as the input ($11)

c) (2) Does either or both of these instructions cause an overflow exception? If so, identify the instruction(s).

The first instruction produces an overflow, which is why the value wraps around from a large negative to a large positive on the addition of a negative number. The second instruction does not produce an overflow.

15. (6) Suppose that a hardware error, hack attack, or bit flip due to cosmic radiation inverts bit 20 within the machine instruction for bgez $4,exit. Bit 20 is the only bit affected. Would the modified 32-bit machine instruction still correspond to a valid instruction? If so, what is the assembly language instruction, when translated to machine code, corresponds to the modified 32-bit pattern?

The instruction would still be valid, however the instruction would not point to exit. Instead the instruction would point to some value in memory other than exit, depending on the initial value of the offset.