CS 0447 - Fall 2009 - Exam 1 Sample Problems

The questions below are to help study for the exam. They will give you an idea of the kinds of questions and knowledge that is expected. The programming problems are intended to give you more examples to gain experience with MIPS. Some of the programming problems are too long for the exam, but you should be familiar with the concepts illustrated (e.g., problems 16-18). It is recommended that you try the problems without looking at the solution. The solution is on the web (including example programs).

1. Convert these binary numbers into hexadecimal:
a) 000111010011
b) 01111110010101101001001100100100
c) 1110111101111010
d) 10111100001111110110

2. Convert these binary numbers into octal:
a) 000111010011
b) 00111111001010110100100110010100
c) 001110111101111010
d) 010111100001111110110

3. Convert these hexadecimal numbers into binary:
a) 0AAF
b) BC790183
c) 6FD23EA5
d) 4D88F072

4. Convert these hexadecimal numbers into octal
a) 0AAF
b) BC790183
c) 6FD23EA5
d) 4D88F072

Programming Problems

Here are several sample problems to test your understanding of the MIPS instruction set. I have marked the problems as easy to hard. Some of the problems are too long for an exam, but you should be familiar with the concepts.

5. Write MIPS code that adds a number in $t1 to the constant 100 and puts the result into $t3. (EASY)

6. Write MIPS code that does the computation:
F = A + (((B - C) + D) - E)
Assume that variable A is in register $t0, B is in register $t1, C is in register $t2, D is in register $t3, E is in register $t4 and F is in register $t5 (the result). (EASY)

7. Write MIPS code that prints "It is negative" when the value in register $t1 is less than 0. (MODERATE)

8. Write MIPS code that prints "The bit is 0" when bit 15 in$t0 is 0 and prints "The bit is 1" when bit 15 in $t0 is 1. (MODERATE)

9. Write MIPS code that sets bits 7 and 9 (makes them 1) in $t0. The code should preserve the contents of all other bits. (EASY)

10. Write MIPS code that clears bits 11 and 13 (makes them 0) in $t0. The code should preserve the contents of all other bits. (EASY)

11. Write MIPS code that prints the value for bits 15, 16, and 17 in $t0. The code should print a single integer number that corresponds to the 3 digit binary number in bits 15, 16, 17. (MODERATE)

12. Write MIPS code that prints the 32-bit value in $t0 as a hexadecimal number. Hint: your program should use a table lookup to convert to characters for the hex number. You will find a loop useful to extract 4 bit quantities from $t0, which are then converted into hex characters. (HARD)

13. Using a shift operation, write MIPS code that multiplies the number in $t0 by 16. (EASY)

14. Using a shift operation, write MIPS code that divides the number in $t0 by 8. (EASY)

15. Using only `ori` and `sll` instructions, put the constant 0x3FF01 into $t0. (EASY)

16. Write MIPS code that prompts the user to enter two numbers, adds the numbers, and prints the result. The prompts should be "Number 1?" and "Number 2?". When the code prints the number, it should print "The result is X", where "X" is the sum. (MODERATE)

17. Write MIPS code that emulates a simple adding machine. The “adding machine” initially has the value 0. Prompt the user to enter a command. A command is a number, where 0 is add, 1 is subtract, 2 is multiply, 3 is divide, 4 is clear, and 5 is exit. When the user enters 0, 1, 2, or 3, prompt the user for a number to add, subtract, multiply or divide from the current value. If the user enters 4, then clear the current value (set it to 0). If the user enters 5, exit the program. Give useful prompts. (HARD)

18. Write MIPS code that finds and prints the smallest number in the integer array "arry". The declaration of the array is shown below. (MODERATE)

```
.data
.align 2
arry: .word  100,10,3,-2,110,0,-50,150,-17,8
```

19. NO PROBLEM PROVIDED.

20. List three embedded computers you used today.

21. List three server software applications that you used today.

22. What are the five classic components of a computer system?

23. Encode the following MIPS instructions into binary. You should use the green card at the front of the book. (On an exam, I will give you the values for the opcode, etc., but you should know the positions of the fields in I-format, J-format, and R-format instructions.)
a) `add $t0,$t1,$t2`
b) `addi $s0,$s1,100`
c) `srl $v0,$v1,17`
d) `beq $t7,$0,L10`
* assume beq instruction address is 0x0040002c
* assume L10 is address 0x0040004c

24. Why does the MIPS instruction set have the lui instruction?

25. Suppose you want to perform the operation "not $t0". Give a MIPS instruction that can perform this operation.

26. NO QUESTION PROVIDED.

27. Consider a base-32 number. We can represent a digit in this number system using a combination of numbers and letters. Answer the following:
a) How many binary digits does each base-32 digit represent?
b) In comparison to hexadecimal, why might this base-32 number system be cumbersome for assembly language programmers?

28. Briefly explain the purpose of the compiler.

29. Aside from specifying instructions, what else might a processor reference manual (PRM) specify for an instruction set architecture?

30. In MIPS, how many general-purpose registers actually allow their values to be changed?

31. What is the value of $t2 after the code below is executed?

```
ori $t0,$0,255
addi $t1,$t0,1
sll $t2,$t1,2
```

32. What is the value of $t2 after the code below is executed?

```
 ori $t0,$0,512
 slt $t1,$0,$t0
 bne $t1,$0,L0
 ori $t2,$0,1024
 j L1
L0: ori $t2,$0,2048
L1:
```

33. What is the value of $t3 after the code below is executed?

```
 .data
 .align 2
 var:   .word 0xF0FFF0FF,0xFF00FF00
 .text
 la $t0,var
 lw $t1,4($t0)
 lw $t2,0($t0)
 xor $t3,$t1,$t2
```

34. Which of the following accesses cause an error due to misalignment?

```
.data
 .align 2
 var:   .word 0x01020304
 .text
 la $t0,var
 lw $t1,0($t0) # access 1
 lw $t1,1($t0) # access 2
 lw $t3,2($t0) # access 3
 lb $t2,0($t0) # access 4
 lb $t2,1($t0) # access 5
 lb $t2,2($t0) # access 6
 lh $t3,0($t0) # access 7
 lh $t3,1($t0) # access 8
 lh $t3,2($t0) # access 9
```

35. What is the value of $t3 after each instruction below is executed? Assume v1 is aligned on a word boundary.

```
.data
 v1:    .byte 0x0,0x1,0x2,0x3,0x3,0x2,0x1,0x0
 .text
 la $t0,v1
 lw $t3,0($t0) # value of $t3?
 lw $t3,4($t0) # value of $t3?
```

36. Suppose $t2=0xFF00080 and the word stored at this address is 0x000000FF. Assume little endian addressing. What is the value of $t3 when the instruction lb $t3,0($t2) is done?

37. Suppose $t2=0xFF00080 and the word stored at this address is 0x000000FF. Assume little endian addressing. What is the value of $t3 when the instruction lbu $t3,0($t2) is done?

38. Consider a function bart. It does not have any arguments. Give a single MIPS instruction that does a function call to bart.

39. Now, consider function homer. It does not have any arguments or return a value. Give a single MIPS instruction that does a function return, assuming the return address is in register $ra.