

EECS 370 Fall 2008

Homework 1

Solution

Assigned: Thursday, September 4, 2008
Due: Tuesday, September 16, 2008 (in class)

Problem 1 (6 points)

Convert the following decimal values to both binary and hexadecimal representations.
Example: $132_{10} == 10000100_2 == 84_{16}$.

SOLUTION:

(1 point for each correct answer)

$196_{10} == 11000100 == 0xC4$

$2807_{10} == 101011110111 == 0xAF7$

$49_{10} == 00110001 == 0x31$

Problem 2 (8 points)

Given the following MIPS assembly code:

```
    add $1, $0, $0
    addi $2, $0, 50
lp:  andi $3, $2, 1
     bne $3, $0, 1
     add $1, $1, $2
     addi $2, $2, -1
     bne $2, $0, lp
```

How many instructions would be executed by a MIPS processor running this code? Also provide a brief one or two sentence explanation of what the code is doing.

Solution:

227 instructions:

2 initial instructions

25 loop iterations of 4 instructions (internal branch taken)

25 loop iterations of 5 instructions (internal branch not taken)

(4 pts for this part)

(Either of the following would be good for 4 remaining points)

Code description:

sum = 0

for (i = 50; i != 0; i--) {

if ((i & 1) == 0) {

sum = sum + i

```

    }
}

```

Text explanation: This computes the sum of the even numbers in the range 1 through 50.

Problem 3 (4 points)

Assume a 64-bit wide RISC machine with 64 registers and 200 distinct instructions. Its branch instruction compares two registers for equality, and uses the 2's complement offset field to compute the PC relative target. What is the branch target range?

Solution:

-2^{43} to $2^{43}-1$

64 bit instruction:

8 bits for opcode (maximum of 256 opcodes)

6 bits for register 1 ($2^6 = 64$ addressable registers)

6 bits for register 2

44 bits left for branch target (44 bits can represent a number between -2^{43} and $2^{43}-1$ in 2's complement)

Problem 4 (12 points)

Most computers represent signed numbers in 2's-complement. Given a 32-bit number, in 2's-complement, write LC-2K8 assembly code that calculates the absolute value. Assume that the number is already stored in register 1 and the absolute value is to be placed in register 2. The original number in register 1 does not need to be preserved. Try to do this in less than a dozen lines of code.

Solution:

```

lw 0 3 mask          # load mask (0x80000000 into register 3)
nand 1 3 4           # no AND available so NAND mask and original,
nand 4 4 4           # then nand the result with itself
beq 0 4 done         # branch if positive
nand 1 1 1           # the number is negative, nand it with itself
lw 0 5 one           # load '1' into register 5
add 1 5 1            # add 1 to register 1 and store result in register 1
done                 add 0 1 2          # move contents of 1 to 2
                    halt
mask                 .fill 2147483648   # hex 0x80000000
one                   .fill 1

```

Many other solutions possible