CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN—MADISON

Prof. Mark D. Hill
TAs Marc de Kruijf & Sanghamitra Roy

Midterm Examination 1
In Class (50 minutes)
Wednesday, February 14, 2006
Weight: 15%

CLOSED BOOK, NOTE, CALCULATOR, PHONE, & COMPUTER.

The exam in two-sided and has **6** pages, including two blank pages at the end.

Plan your time carefully, since some problems are longer than others.

NAME: ____

ID# ____

Problem Number	**Maximum Points**	**Graded By**
1	4	SR
2	3	SR
3	3	SR
4	4	SR
5	4	SR
6	4	MK
7	4	MK
8	4	MK
Total	30	

Problem 1 (4 points)

(a) What is the largest (most positive) integer that can be represented as an *unsigned* integer using 10 bits?

$1111111111_b = (2^{10} - 1)_d = 1023_d$

Full credit for any one of $(2^{10} - 1)_d$ or 1023_d

(b) What is the largest (most positive) integer that can be represented as a *two's complement* integer using 10 bits?

$0111111111_b = (2^9 - 1)_d = 511_d$

Full credit for any one of $(2^9 - 1)_d$ or 511_d

Problem 2 (3 points)

Consider bitwise logical operations:

(a) Compute `0011 AND 1001`

0001

(b) Compute `0011 OR 1001`

1011

(c) Compute `NOT 1001`

0110

Problem 3 (3 points)

Convert the number `-33` (base ten) into two's complement representation with 8 bits.

$+33_d = 00100001_b$

$-33_d = 11011110_b + 1_b = 11011111_b$

Problem 4 (4 points)

Consider the 8-bit binary bit pattern `11010111`. What is its decimal (base ten) value if the bit pattern is interpreted as:

(a) An unsigned integer

$11010111_b = 128 + 64 + 16 + 4 + 2 + 1 = 215_d$

(b) A two's complement integer

$11010111_b = -(00101001)_b = -41_d$

A common mistake in this problem was calculating signed magnitude representation (-87) instead of two's complement.

Problem 5 (4 points)

(a) Add the following 5-bit two's complement binary numbers: `01110 + 10110`. Express your answer in 5-bit two's complement. Please indicate if there was an overflow.

00100

There is no overflow. If nothing is mentioned about overflow, no overflow is assumed and full credit given. Common mistake was indicating overflow.

(b) Add the following 5-bit two's complement binary numbers: `11101 + 11111`. Express your answer in 5-bit two's complement. Please indicate if there was an overflow.

11100

There is no overflow. If nothing is mentioned about overflow, no overflow is assumed and full credit given. Common mistake was indicating overflow.

Problem 6 (4 points)

(a) Convert the hexadecimal number (base sixteen) number `6A1F` into binary

$6_{16} = 0110_2$, $A_{16} = 1010_2$, $1_{16} = 0001_2$, $F_{16} = 1111_2$ Answer = 0110101000011111

If you were off by one on any one of the above, you still received full credit.

(b) Convert the binary number `10011101` into a hexadecimal number (base sixteen)

$1001_2 = 9_{16}$, $1101_2 = D_{16}$ Answer = 9D

Again, if you were off by one on any one of the above, you still received full credit.

Problem 7 (4 points)

(a) Why do most computer represent numbers in binary (base two) rather than decimal (base ten)?

It is much easier for computers to distinguish between the presence and the absence of voltage than to distinguish between multiple voltage levels. Answers dealing with on/off transistor states and component cost and/or efficiency were also accepted.

(b) What is the name given to the hardware/software interface?

Instruction Set Architecture (ISA). Architecture was only half-credit. If you were visibly confused by the slash (thinking "/" was "or") I tried to be lenient and accepted ISA as either the hardware or software answer.

Problem 8 (4 points)

(a) Given an example of a number that can be represented in 32-bit IEEE 754 floating-point that cannot be represented as a 32-bit two-complement integer. Explain why.

Two examples: 0.5 and 2^{32}. The example should either demonstrate the fractional property or exponent property (or both) of floating-point, and how that allows for a greater range of values. If you were confused by the typo "Given" and did not provide an example, I accepted only an explanation. (I made this decision late and went back and fixed old exams. If I somehow missed yours by accident, please see me.)

(b) Computer C1 has an instruction for addition but no instruction for multiplication. Computer C2 has instructions for both addition and multiplication. Can computer C2 solve more problems than computer C1? Why or why not?

No. Computer C1 can implement multiplication by doing a series of additions. For example, $5 \times 7 = 7 + 7 + 7 + 7 + 7$. Hence, computer C1 can solve the same number of problems as C2.