NATIONAL UNIVERSITY OF SINGAPORE

SEMESTER EXAMINATION FOR THE DEGREE OF B. SC.

SEMESTER 1

CZ101 BASIC COMPUTER ARCHITECTURE AND ASSEMBLY PROGRAMMING

October/November 1996

Time allowed: 2 hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **SEVEN (7)** questions and comprises **THREE (3)** printed pages.
2. Answer all questions.
3. Candidates may use calculators. However, they should lay out systematically the various steps in the calculations.

1. Answer briefly:

   (a) What are the five classic components of a computer?

   (b) Draw a diagram to show the memory hierarchy. Indicate the hardware components in the diagram.

10 marks

2. What are the contents in register \$2 to \$5 and new contents in memory after the execution of the following instructions?

```
        addi  $4, $0, 1000
L1:     sub   $2, $0, $4
        slti  $3, $2, -1
        sw    $3, 4($4)
        beq   $2, $3, L1
        lw    $5, 1004($0)
```

12 marks

3. The following VAX instruction moves the negated value of register \$5 into register \$3:

```
mnegl  $3, $5      #  $3 = - $5
```

   What is the single MIPS instruction, or if it cannot be represented in a single instruction, the shortest sequence of MIPS instructions, that performs the same operation?

5 marks

4. The MIPS assembly instruction (pseudoinstruction)

```
mul  $2, $4, $5     #  $2 = $4 * $5
```

   computes the product of 32-bit two's complement numbers, and puts the result in a 32-bit register.

   (a) Write a sequence of MIPS *machine* instructions that do the same as the above assembly instruction.

   (b) Write a sequence of instructions to detect overflow of the result in \$2, so that \$3 contains 1 if overflow occurs, and 0 if overflow does not occur.

15 marks

5. Carry out integer division of 4-bit unsigned binary numbers $\texttt{1111}_{two} \div \texttt{0110}_{two}$, follow step by step the third division algorithm: 4-bit divisor is in $D = 0110$, and an 8-bit register R contains initially the dividend 0000 1111. R_h denotes the upper half of R, and R_0 is the 0-th bit of R. Repeat 4 times:

   1. Shift R left 1 bit, $R = R \ll 1$.
   2. Subtract D from the upper half of R, $R_h = R_h - D$.
   3. If $(R \geq 0)$ $R_0 = 1$, else $R_h = R_h + D$.

   What are the decimal values for the quotient and remainder?

15 marks

6. In the IEEE single-precision floating-point format, the most significant bit (bit 31) is the sign bit s, followed by 8 bits for the exponent e, and the rest of 23 bits for the significand,

$$f = b_{-1}\, 2^{-1} + b_{-2}\, 2^{-2} + b_{-3}\, 2^{-3} + \cdots.$$

The value of a floating-point number is

$$(-1)^s \times (1 + f) \times 2^{e-127}.$$

(a) Find the 32-bit floating-point IEEE representation of a decimal fraction 0.40625.

(b) Two floating-point numbers x and y are given below as bit patterns. What values do they represent?

```
1 01111110 00000000000000000000000
0 10000000 10000000000000000000000
```

(c) Compute $x \times y$, follow roughly the steps a computer might take. Show result as a bit pattern.

18 marks

7. Translate the following C function into a complete MIPS assembly language procedure. The procedure takes \$4 and \$5 as the first two arguments and returns a value in \$2.

```
int select(int a[ ], int b[ ])
{
   int i, count = 0;

   for (i = 0; i < 100; ++i) {
      if ( a[i] > 0) {
         b[i] = a[i] + 1;
         ++count;
      }
   }
   return count;
}
```

Note that the `select()` function loops over the first 100 elements in array `a[ ]`. If an element is positive, the incremented value is stored at the corresponding element in array `b[ ]`. The occurrences are counted and the total is returned. A table of MIPS instruction set is on next page.

25 marks

— End of the Paper —

CZ101, 1996, Answers

1: (a) The five classic components of a computer are control, data path, memory, input, and output.

(b) Memory hierarchy from small to large (fast to slow): registers, cache memory, main memory, and external hard disk (virtual memory).

2: $\$4 = 1000$, $\$2 = -1000$, $\$3 = 1$, $\$5 = 1$, memory$[1004] = 1$.

3: The VAX `mnegl` is equivalent to MIPS instruction

```
sub  $3, $0, $5    # $3 = $0 - $5 = - $5
```

4: (a) The assembly instruction is equivalent to

```
mult  $4, $5     # machine multiply
mflo  $2         # lower half to $2
```

(b) Overflow does not occur if register HI is 0, and the sign of LI is positive, or HI is `0xFFFFFFF` and the sign of LI is negative. Overflow occurs otherwise. In other words, HI is a sign extension of the LO.

```
mult $4, $5
mflo $2         # the lower half result
mfhi $4         # the upper half
sra  $6, $2, 31 # sign extension of lo
addi $3, $0, 0  # assume no overflow
beq  $4, $6, L  # no overflow if hi is a sign extension
addi $3, $0, 1  # overflow occurs otherwise
L:
```

5: Here is the result

Iteration	step	R
0		0000 1111
1	step 1	0001 1110
	step 2	1011 1110
	step 3	0001 1110
2	step 1	0011 1100
	step 2	1101 1100
	step 3	0011 1100
3	step 1	0111 1000
	step 2	0001 1000
	step 3	0001 1001
4	step 1	0011 0010
	step 2	1101 0010
	step 3	0011 0010

So the quotient is $\texttt{0010}_{two} = 2_{ten}$, remainder is $\texttt{0011}_{two} = 3_{ten}$.

6: (a) First, write 0.40625 in binary.

```
0.40625 * 2 = 0.8125   -> 0
0.8125 * 2 = 1.625     -> 1
0.625 * 2 = 1.25       -> 1
0.25 * 2 = 0.5         -> 0
0.5 * 2 = 1.0          -> 1
```

In binary, the number is `0.01101` or 1.101×2^{-2}. So the IEEE representation is

```
0 01111101 10100000000000000000000
```

(b) $(-1) \times 2^{126-127} \times 1.0000... = -0.5$. $2^{128-127} \times (1 + 1/2) = 2 + 1 + 1/2 = 3.0$

(c) A negative number times a positive number, so the sign bit is 1. The product is $(-1+1) = 0$ for the exponent and $1.0 \times 1.1 = 1.1$ for the significand. Bit pattern:

```
1 01111111 10000000000000000000000
```

7: Here is the assembly program

```
        .text
        .globl select
select:
        addi $7, $0, 0     # for index i
        addi $8, $0, 100   # for limit
        addi $2, $0, 0     # for count
loop:   lw   $3, 0($4)     # load a[i]
        slt  $6, $0, $3    # $6 = (0 < a[i])
        beq  $6, $0, L1    # goto L1 if (a[i] <= 0)
        addi $3, $3, 1     # a[i] + 1
        sw   $3, 0($5)     # b[i] = a[i] + 1
        addi $2, $2, 1     # ++count
L1:     addi $4, $4, 4     # address of next element in a[]
        addi $5, $5, 4     # addr of next element in b[]
        addi $7, $7, 1     # ++i
        bne  $7, $8, loop  # loop until i == 100
        j    $31
```