COE 301 / ICS 233

Computer Organization

Midterm Exam – Term 172

Saturday, March 24, 2018

10:00 am – 12:00 noon

Computer Engineering Department

College of Computer Sciences & Engineering

King Fahd University of Petroleum & Minerals

Student Name: ID:

|  |  |  |
| --- | --- | --- |
| 🞏 Dr. Aiman El-Maleh | 🞏 COE 301 | 🞏 ICS 233 |
| 🞏 Dr. Marwan Abu-Amara | 🞏 COE 301 | 🞏 ICS 233 |
| 🞏 Dr. Muhamed Mudawar |  | 🗹 ICS 233 |

|  |  |  |  |
| --- | --- | --- | --- |
| Q1 | / 19 | Q2 | / 17 |
| Q3 | / 07 | Q4 | / 14 |
| Q5 | / 07 | Q6 | / 16 |
| Total |  / 80 |

Important Reminder on Academic Honesty

Using unauthorized information or notes on an exam, peeking at others work, or altering graded exams to claim more credit are severe violations of academic honesty. Detected cases will receive a failing grade in the course.

**Q1. [19 points] Fill-in the Blanks**

1. (1 point) In addition to being space and time efficient, programming in assembly language has the advantage of ­­­­­­­­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
2. (1 point) In addition to faster program development and maintenance, programming in high-level language has the advantage \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
3. (1 point) The instruction set architecture of a processor provides an interface between \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
4. (1 point) Assuming **Array** is defined as shown below:

**Array: .word 10, 11, 12, 13, 14**

The content of register **$t1** (in hexadecimal) after executing the following sequence of instructions is \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

**la $t0, Array**

**lw $t1, 4($t0)**

1. (2 points) Write a minimum sequence of MIPS basic instructions to implement the pseudo instruction: **bgt $s1,$s2,Next**
2. (2 points) Write a minimum sequence of MIPS basic instructions to implement the pseudo instruction: **andi $t0,$t0,0x12345678**
3. (1 point) Assuming that **$a0** contains an Alphabetic character (uppercase or lowercase), write a MIPS instruction that will make the character stored in **$a0** always lower case. Note that the ASCII code of character **'A'** is **0x41** while that of character **'a'** is **0x61**.
4. (3 points) The following is a partial MIPS assembly language code:

|  |  |  |
| --- | --- | --- |
| **Address** | **Label** | **Instruction** |
| **0x00400000** |  | **bgtz** | **$a1, loop** |
|  |  |  | **. . .** |
| **0x00403000** | **loop:** | **addu** | **$a0, $a1, $v0** |
|  |  |  | **. . .** |
| **0x00410000** |  | **bne** | **$a0, $zero, loop** |

Calculate the 16-bit immediate value (in hexadecimal) for **loop** in the **bgtz** instruction.

Calculate the 16-bit immediate value (in hexadecimal) for **loop** in the **bne** instruction.

1. (2 points) Given that **Array2** is defined as shown below:

**Array2: .byte -1, 2, -3, -4, -5, 6**

After executing the following sequence of instructions, the content of the two registers (in hexadecimal) is **$t1=**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and **$t2=**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

**la $t0, Array2**

**lb $t1, 2($t0)**

**lhu $t2, 2($t0)**

1. (2 points) Given the following data definitions, and assuming that the first variable **X** starts at address **0x10010000**, then the addresses for variables **Y** and **Z** will be \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

**.data**

**X: .byte 10, 11, 12**

**Y: .half 13, 14, 15**

**Z: .word 16, 17, 18**

1. (3 points) Write a minimum sequence of MIPS basic instructions to multiply the **signed** integer value of register **$t0** by **15.25** without using multiplication instructions. Put the final integer result in **$t0**. For example, if the initial value of **$t0** is **5** then the final value will be **76**. The additional fraction is truncated.

**Q2. Floating-Point [17 points]**

1. (3 points) Find the decimal value of the following single-precision float:

|  |  |  |
| --- | --- | --- |
| **S** | **Exponent** | **Fraction** |
| **1** | **1000 1000** | **111 1100 1101 0000 0000 0000** |

1. (4 points) Find the **normalized** IEEE 754 single-precision representation of **+59.625**.
2. (2 points) Show the IEEE 754 representation of **+Zero** and **-Infinity** for single precision:
3. (2 points) Find the approximate decimal value of the largest positive **denormalized** float for single precision.
4. (6 points) Given that **A** and **B** are single-precision floats, compute the difference **A–B**. Use rounding to **nearest even**. Perform the operation using **guard**, **round** and **sticky** bits.

**A = +1.111 0000 0000 1111 1100 0001 × 2-4**

**B = +1.000 1111 1111 0000 0000 1111 × 2+3**

**Q3. [7 points] Translate Nested IF statements**

Using minimal number of instructions, translate the following nested IF statements into MIPS assembly code. Assume that variables **a**, **b**, **c**, and **d** are **signed integers** stored in registers **$s0**, **$s1**, **$s2**, and **$s3**, respectively. If needed, **you can use pseudo-branch instructions only**.

**if ( ((a > b) || (c <= d)) && (a == c) ) {**

 **if (c != d) a = b + c;**

 **else c = b – 2;**

**}**

**Q4. [14 points] Translate a Recursive Function**

Translate the following high-level recursive function **freq** into MIPS assembly code. The **freq** function counts the number of times an integer **i** appears in an array **A** of **n** integers. The array **A** is already in memory. The function parameters are passed in registers **$a0**, **$a1** and **$a2**, respectively. The **freq** function returns the result in register **$v0**. Use **MIPS convention** in saving and restoring **only the necessary register(s)** in the recursive function. Your MIPS implementation of the **freq** function **must be recursive**. Add comments to explain the utilization of registers.

**int freq(int A[], int n, int i) {**

 **if (n == 0) return 0;**

 **int j = 0;**

 **if (A[n-1] == i) j = 1;**

 **return (j + freq(&A[0], n-1, i));**

**}**

**Q5. [7 points] Compute the Sum of Decimal Digits**

Write a MIPS function **sum\_digits** thatcomputes and returns the **sum of decimal digits** in an **unsigned integer**. For example, the sum of decimal digits for **1536** is **1+5+3+6 = 15**. The function **sum\_digits** receives the unsigned integer argument **in binary** in register **$a0**. For example, **1536 = (0000 … 0110 0000 0000)binary**. It should extract the decimal digits, compute, and return their sum, **also in binary**, in register **$v0**. Hint: divide the unsigned integer by **10** to extract the decimal digits.

**Q6. [16 points] Write Loops to Traverse a Matrix**

Given that **M** is a square matrix of *N*×*N* integers (*N* rows by *N* columns), which is already read and initialized in memory. The **starting address** of matrix **M** is stored in register **$a0**, and *N* is stored in register **$a1**. This is always the case for parts (**a**) and (**b**).

**a)** (7 points) Write MIPS code to compute the sum of all *N* elements of row *i*, where *i* < *N*. The value of *i* is stored in register **$a2**. The sum should be computed in **$v0**. Add comments to explain the utilization of registers. **Do NOT use pseudo-instructions**.

**b)** (9 points) Write MIPS code to locate the maximum element in column *j*, where *j* < *N*. The value of index *j* is stored in **$a2**. The **maximum** **unsigned value** of column *j* should be computed in **$v0** and its corresponding row index should be computed in **$v1**. Add comments to explain the utilization of registers. **You may use pseudo-branch instructions**.

















