

Homework #3

Due: at the start of lecture on Thursday 05 July 2007.

Write NEATLY or TYPE your answers.

Please Xerox your homework before submitting it, so you have verification in case any of your paper(s) is(are) missing.

Part I – Programming Problems

1 (10 pts) (a) Write a MIPS program that computes the cube of the difference between two integers x and y :

$$\text{cubed difference} = (x - y)^3.$$

Show how this program would be implemented as a MIPS instruction `cdf $t2, $t0, $t1`. Diagram the algorithm and ALU (block diagram only) that would be used to implement this instruction, and identify the computational or data-movement bottlenecks that exist in the ALU or algorithm. **Be sure to use the MIPS `mfhi` and `mflo` instructions.**

2 (20 pts) (a) Use the MIPS program and new MIPS instruction `cdf` developed in Problem 1, above, to write another MIPS program to compute the *mean-cube difference* of two n -element arrays **a** and **b** of floating point numbers according to the following equation:

$$\text{MCD}(\mathbf{a}, \mathbf{b}) = 1/n \cdot \sum_{i=1}^n (\mathbf{a}(i) - \mathbf{b}(i))^3$$

(b) Analyze the execution time of the program you wrote in 2a), above, according to the following table of CPI's, assuming a 4.3 GHz clock rate:

Instruction	Cycles
mfc0, mfhi, mflo	1
slt, slti, sltiu, j	2
lw, sw, bne, beq	3
jr, jal	4
add, addi, sub, addu, addiu, subu	6
mult, multu, div, divu	11

Include in your analysis a graph of execution time as a function of the array length n , for n equal to the first 20 powers of 2. You can produce this graph in MS-Excel, if you wish.

3 (10 pts) Given the following 32-bit patterns:

- #1: 1101 0101 1001 1011 0101 1100 1011 0010
- #2: 1100 1110 1001 1010 0111 0101 0100 0001

Derive, *and show how you derived*, the number or instruction that each bit pattern represents, assuming that they are:

- (a) a two's complement integer
- (b) an unsigned integer
- (c) a single precision floating-point number
- (d) a MIPS instruction – if not a valid instruction, then explain why
- (e) a 4-character (ASCII) array (8 bits per character, standard ASCII)

Show how you arrived at each answer - *show your work!* If you do not know what standard ASCII encoding is, then search Google using keywords such as ASCII code encoding.

4 (10 pts) Write a MIPS program that implements Booth's algorithm for multiplication (as we discussed in class – refer to viewgraphs, Lecture #13) for 32-bit integers only (no floating point). Store the multiplicand, multiplier, and product in memory and use the appropriate load/store commands to input/output them to/from registers.

You are strongly encouraged to use the SPIM program to verify that your program works. After you get your MIPS program working, modify it so that it pushes onto the stack the multiplier, multiplicand, and all partial and accumulated products, together with the result (final product), at the appropriate times (e.g., as they are read from memory, and as the partial products are generated).

Then, add a procedure or some in-line code to your MIPS program that can print these values by popping them off the stack and uses the system call (discussed in Appendix A of your text) to print out the multiplication problem as follows (example for a four-bit multiplication):

```

                                0 0 1 0   N-bit multiplicand
                                1 0 1 1   N-bit multiplier
                                -----
0 0 1 0 1 0 1 1   2N-bit product register for
iteration #1
    etc...        more 2N-bit product register
contents...
-----
    result        final contents of 2N-bit product
register

```

Part II - Extra Credit

5 (20 pts) Write one small MIPS program to implement each of the three algorithms (v.1, v.2, and v.3) [so, you need to write three programs total] for *unsigned multiplication* that we discussed in class (see viewgraphs, Lecture #13). Use the shift instructions native to MIPS to operate the shift registers. *You may not employ any kind of multiplication instruction or pseudo-instruction in any program.* Analyze your programs by making a table of each instruction type and its frequency of occurrence. Determine which program would run fastest using the table of CPIs from Problem 2, above. *Show all work to get full credit.*

6 (10 pts) Compare and contrast the hardware requirements, advantages, and disadvantages of the ALUs that implement each of the three algorithms (v.1, v.2, and v.3) for unsigned multiplication that we discussed in class. You can use the results of Problem 5, above, to support your assertions.

*** 7 (25 pts)** Write a MIPS program to perform *double precision floating point multiplication* using only *integer operations* (no `mul.s` or `mul.d` instructions). Assume that double-precision floating point numbers have one sign bit, 31 bits for the exponent, and 32 bits for the significand, and are manipulated just like IEEE 754 floating point numbers (except for the size of the exponent and significand). You need to follow all procedure calling conventions used in MIPS (i.e. stack frame allocation, register saving, etc.). In SPIM, use a main procedure to test your program. Instead of using registers \$t4 -

\$t7 to store the 2 64-bit operands, `main` will pass them as arguments. The first 64-bit number will be stored in registers `$a0-$a1` and `$a2-$a3` and the 128-bit result will be pushed on the stack before exiting your procedure. The base and offset for the 128-bit result will be returned in registers `$v0` and `$v1`, respectively.

Please email the TAs if you have a question about the homework.
