SCC.150 - MIPS/Assembly Week 14 practical

C. Rotsos, I. Aref

SCC - Lancaster University

Week 14

Outline

1 Introduction

2 Ethiopean Multiplication

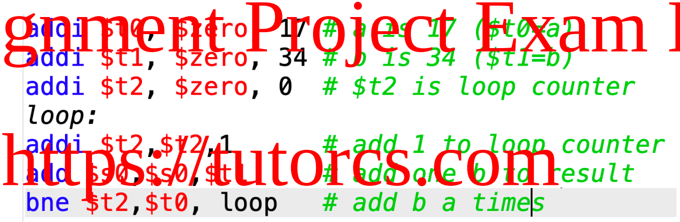
3 Task

4 Marking

MIPS arithmetics

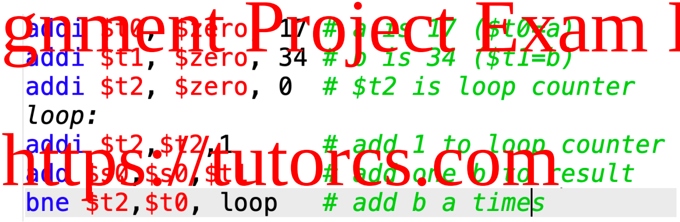
- MIPS ISA aritmetic support is limited to adding.
- CoProcessor1 iadvanced arithmetic operation (e.g. multiplication) and floating point number support.
- Support for foating point operations inreases the complexity and production cost of the CPU.
- Can you build an assembly program to multiply numbers using simple arithmetic operations?

A simple solution



```
addi $t0, $zero, 17 # a is 17 ($t0=a)
addi $t1, $zero, 34 # b is 34 ($t1=b)
addi $t2, $zero, 0  # $t2 is loop counter
loop:
addi $t2,$t2,1      # add 1 to loop counter
add $s0,$s0,$t1     # add one b to result
bne $t2,$t0, loop   # add b a times
```

A simple solution



```
addi $t0, $zero, 17 # a is 17 ($t0=a)
addi $t1, $zero, 34 # b is 34 ($t1=b)
addi $t2, $zero, 0  # $t2 is loop counter
loop:
addi $t2,$t2,1      # add 1 to loop counter
add $s0,$s0,$t1     # add one b to result
bne $t2,$t0, loop   # add b a times
```

- Add a times number b to the result.
- Takes a clock-cycles to compute result.
- Awful performance for large numbers.
- Can we do better?

Algorithm

- ***Ethiopian multiplication*** is an algorithm to multiply positive integers using bit-shifts and additions.
- Algorithm:
  1. If number a is odd, add value b in result.
  2. Half number a, double number b.
  3. If a is not 1, got to step 1.
  4. The result contains $a * b$.
- BBC mini documentary on Ethiopian multiplication: `https://www.bbc.co.uk/programmes/p00zjz5f`

Ethiopean Multiplication Example

a	b
17	34
8	~~68~~
4	~~136~~
2	~~272~~
1	544
	578

Hints

- The previously outlined Ethiopian Multiplication can be implemented using assembler instructions.
- `sll` and `srl` are shift operations which can be used to implement doubling and halving (multiplication by 2, division by 2).
- `and` can be used to implement a test to check if a number is odd or even

The task

A program for Ethiopian Multiplication should be constructed. The program should read the two numbers a and b to be multiplied from memory. Then a function (procedure) should be called which receives a and b as parameters. The function then carries out the multiplication and returns the result. The result is then stored in memory and the result is printed on screen using a ***syscall***.

Program details

- items a and b should be read from a two-integer array located in address ***0x10010004***.
- The main program should pass the address of the input array to the procedure using register $a0.
- After reading a and b from memory a procedure should be called which implements the Ethiopian multiplication.
- store the multiplication result at address ***0x10010000***.
- The procedure should return 1 using register $v0 to the main program to identify if the multiplication result in an overflow.
- In addition, the multiplication result should be printed on screen by the main program using a ***syscall***.

Coursework Task List

- Implement a correct multiplication algorithm.
- Read and store parameters in memory.
- Implement multiplication in a procedure.
  - Ensure register state is handled correctly.
  - Access and return from and to the function using the appropriate instructions.
  - Use correct registers to pass and read information to the procedure.
- Print a result using a syscall.
- Validate input (Check if it longer than 16-bits).
- Support negative numbers.

Marking and Feedback

- To get marked you have to: (i) Moodle submission by 20/2/2020, (ii) Code Exam on week 18.
- On week 18 you will sit together with a TA/Academic during you designated lab session and discuss you code.
- Examiners will ask you about your code and you will loose marks if you cannot explain your code.
- Need more help: Tuesday 14:00 — 16:00 in Lab B070.