**Homework 3 (Due: Oct 30)**

**Instructions**

* Complete the following questions to the best of your ability.
* Answers should be clear, concise, and justified with work.
* Please write your name and NetID on the hardcopy of your solution, and bring it to the lecture on the due day. Please submit your solution before the lecture starts.

# **Collaboration**

* Professor/TAs
  + You **may** discuss any content with any professor or TA.
* Students
  + You **may** discuss high-level concepts, techniques, jargon, keywords, related problems, or course content that is relevant to the material.
  + You **may not** discuss particular solutions to any of the questions.
  + If you have substantial conversations with any students, please note their name and, if you feel it necessary, the extent of your collaboration.
* Outside Sources (Internet, books)
  + You **may** use external references for any course content.
  + You **may** use an external tool to verify your solutions where appropriate, but not to generate solutions to any questions.
  + List any outside sources that you use. Formal citations are not necessary; links are fine.

# 

# **Question 1 (30 points)**

Implement the following C code in MIPS assembly following the framework and hints provided below.

int func (int n) {

if (n==1)

return 1;

else

if(n == 2)

return 2;

else

return n + fun(n-2);

}

Fill in the blanks of the partial answer. Each blank is one instruction only.

func: addi $sp, $sp, -12 #make room to save 3 values on stack

\_sw $ra, 8($sp)\_\_\_\_\_\_\_\_\_\_\_\_\_\_ #push $ra on stack

\_sw $s0, 4($sp)\_\_\_\_\_\_\_\_\_\_\_\_\_\_ #push $s0 on stack

sw $a0, 0($sp) # push $a0 on stack

#the following instructions are for the if-else-else statement

addi $t0, $0, 1 #set $t0 to 1

\_bne $t0, $a0, base\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # if n != 1, jump to base

\_addi $v0, $0, 1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ # else return 1

j rtn

base: addi $t0, $0, 2 #set $t0 to 2

\_bne $t0, $a0, gen\_\_\_\_\_\_\_\_\_\_\_ #if n != 2, jump to gen

add $v0, $0, $t0 #else return 2

j rtn

gen: addi $t1, $a0, 0 #copy n to $t1

\_subi $t1, $t1, 2\_\_\_\_\_\_\_\_\_\_ #calculate n-2

jal func #call func(n-2)

add $v0, $v0, $t1 #calculate func(n-2) + n

# the following instructions restores register and prepare to return

rtn: lw $a0, 0($sp) #pop $a0

\_lw $s0, 4($sp)\_\_\_\_\_\_\_\_\_\_\_\_\_ #pop $s0

\_lw $ra, 8($sp)\_\_\_\_\_\_\_\_\_\_\_\_\_ #pop $ra

addi $sp, $sp, 12 #restore $sp (top of stack)

jr $ra

# **Question 2 (40 points)**

Two C-based versions of the function find are provided below – an array-based version and a pointer based version. You should assume that (i) find is a leaf procedure and (ii) no $s registers, the return address register, etc. need to be saved to the stack. Note that the array index is returned.

Array-based version: Pointer-based version:

int find(int a[], int n, in x){ int find(int \*a, int n, int x){

int i; int \*p;

for(i = n; i >= 0; i--){ for(p = a+n; p != a; p--){

if(a[i] == x) return i; if(\*p == x) return p-a;

} }

return -1; return -1;

} }

Part A (20 points):

Translate the array-based version into MIPS assembly.

Part B (20 points):

Translate the pointer-based version into MIPS assembly.

# **Question 3 (30 points)**

Assume that logic blocks needed to implement a processor’s datapath have the following latencies:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| I-Mem | Add | Mux | ALU | Regs | D-Mem | Sign-Extend | Shift-Left-2 | FF |
| 200ps | 70ps | 15ps | 90ps | 95ps | 250ps | 15ps | 15ps | 15ps |

1. (10 points) If the only thing we need to do in a processor is to fetch consecutive instructions, what would the cycle time be?
2. (10 points) Consider the single-cycle datapath. What would the cycle time be (not considering the timing requirement of control ROM/random logic)?
3. (10 points) Consider the alternative multi-cycle datapath. What would the cycle time be?