

Homework 4

Due: 14 Oct 2009 1:55pm

All homeworks are due at 1:55pm in the **CS 31 bin** on the second floor. No late homeworks are accepted.

Please include your *login name* on each piece of paper you hand in, and please staple your pages together before handing in.

Some of the problems on this homework must be handed in electronically. Please do each problem in a separate file. Run `cs031_handin hw04` in the directory where your files are stored to hand in. You will be prompted for the name of each file.

Problem 4.1

The following program is being executed on a Moon-1 computer.

```
        lim 3
        sad 3
        lim 1
        sad 2
        lim 0
        sad 1
        lad 3
        sad 0
start:  lad 0
        sou
        sub 2
        sad 0
        jz  end
        lad 1
        jz  start
end:    lim 4
        sou
```

Trace through its execution, and after each instruction has been executed, fill out the table the next page with the current values in the accumulator, the four slots in memory (addresses 0 through 3), and the output register.

Use an **x** in an entry to denote that this value hasn't been written to yet, and thus is at some arbitrary initial state.

You may find that some lines of code are executed more than once due to jumping around. List these instructions as they are executed (this means that your table might have one instruction listed several times). To make your life easier we've given you exactly the number of rows you need and filled out the first two instructions for you:

Time	Instruction	Acc	a0	a1	a2	a3	Out
1	lim 3	3	x	x	x	x	x
2	sad 3	3	x	x	x	3	x
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							

Problem 4.2

In Moon-1's single-cycle implementation, assume that the instructions take the following time to propagate.

Instruction	Propagation Time
<code>lim</code>	20 ns
<code>sad</code>	40 ns
<code>sub</code>	50 ns
<code>sou</code>	20 ns

For Moon-2's multiple-cycle implementation, recall that these instructions are broken down into multiple sub-instructions. Assume that the slowest sub-instruction, loading from memory, propagates in 20 ns.

You will want to refer to the lecture slides on Moon-2 for the specifics on how the instructions are broken down into sub-instructions and how long they take.

- What is the optimal length of the clock cycle for Moon-1? For Moon-2? Explain.
- How long will it take to execute a single `lim` instruction on Moon-1? A single `sad` instruction? `sub`? `sou`? How about on Moon-2? Explain.
- Consider the following chunk of Moon assembly.

```
lim 4
sad 01
lim 5
sub 01
sou
```

How long will it take to execute the above code on Moon-1? On Moon-2?

Show your work.

- Compare the execution times for Moon-1 and Moon-2. Explain why your results are or are not what you would expect.

Problem 4.3

Harry is late for a Quidditch match, and he's being emo because Moon 2 is not as fast as he would like it to be before he has to go. He asks you for help, and though it violates collaboration policy and if you are caught, points will be deducted from your house, you break the rules with him (Harry Potter, breaking the rules?! When did THAT ever happen?).

Assume this is the timing table for all Moon-2 instructions, and that the current design uses a clock cycle of 100 ns. Note that propagation time is independent of the clock cycle:

Instruction	Propagation Time	Frequency
LAD	100 ns	30%
JZ	50 ns	30%
SUB	130 ns	15%
SAD	100 ns	15%
LIM	50 ns	10%

You and Harry are a bit flummoxed, so you call in the advice of your old CS31 groundskeeper, Hagrid.

He says, "Yer current design uses a clock cycle of 100 ns, ammiright? We should reduce the clock cycle to 50 ns an' have some instructions take more cycles. JZ, SUB, and LIM could all be completed faster, while the other instructions would take no additional time."

For this problem, assume that you can slice up the propagation time of these instructions in anyway you want, and that reducing the clock cycle time is not cost-free (imagine that it costs more money to lower the clock cycle).

- How big a penalty would using Hagrid's suggestion incur as opposed to a perfect variable clock? ¹ Show your work.
- Can you come up with a clock cycle that is better than Bob's suggested 50 ns? Justify why it works better.

¹Recall from lecture that a perfect variable clock takes only as much time per cycle as needed by the current operation. Thus, it exhibits theoretical best performance.