ECE M116C - CS M151B Computer Architecture Systems - UCLA

# Quiz 1 (Upload it to Gradescope) - Open book and note

Fall 2022

Don't forget to sign in before you leave.

Q1 [50 points] Assume the CPU is a **single-cycle design**, and the load word (`LW`) is the slowest instruction in the set. The **assembly code** is given below.

```
           ADDI x2, x0, 0
           ADDI x5, x0, 0
           ADDI x7, x0, 1000
LOOP:      LW x3, x2, 0
           ADD x5, x5, x3
           ADDI x2, x2, 4
           BNE x2, x7, LOOP
           SW x2, x5, 4
```

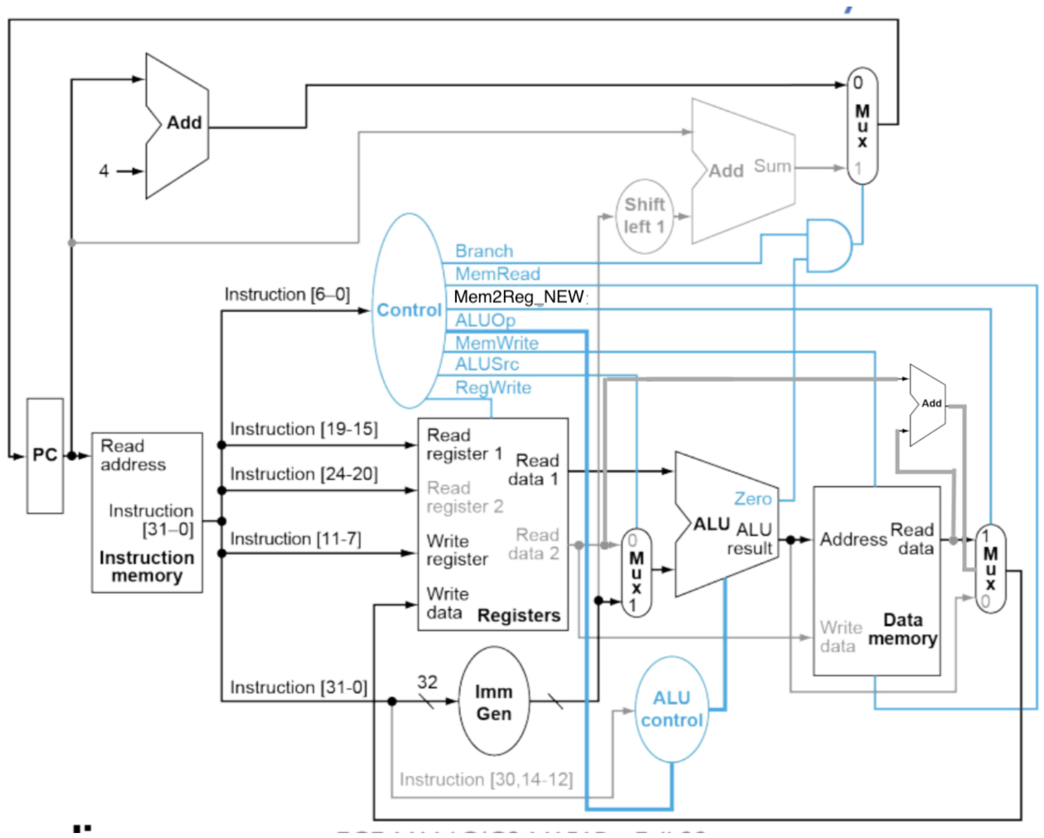
a) Assume that we want to create a new instruction called LW-ADD such that it loads data first and then add it to another register (no immediate is needed for the address). i.e.,

```
LWADD rd,rs1,rs2  → rd = rs2 + Mem[rs1]
```

Draw a datapath+controller that supports this instruction (in addition to the 10 instructions discussed in the class). Your drawing does not need to be pretty(!), and you can only show the details that are relevant to your design.

```
New Load-Add instruction:  LWADD rd,rs1,rs2  → rd = rs2 + Mem[rs1]
```

The new instruction will have its own opcode, and with that the controller can enhance the MemtoReg so that now accommodates a new input to the MEM-WB MUX. An extra adder was added to the data path that adds the RS2 with the data read from the memory. When Mem2Reg_NEW = 2 (0b10) then it selects the new adder's output and forwards it to the write-back stage.



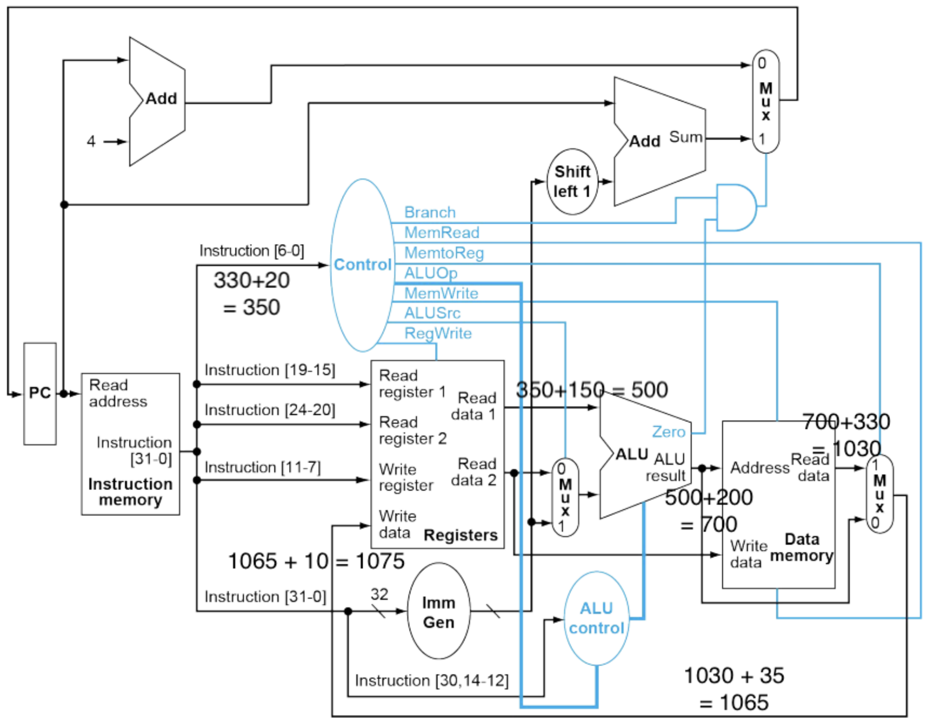
b) What would be the new cycle time of the processor after your implementation of load-add instruction? The latencies of the datapath components are given in the **table** below.

| Mem (I and D) (read) | Reg. File (read) | Reg. File (write) | Mux (any size) | ALU | Gate (any) |
|---|---|---|---|---|---|
| 330 ps | 150 ps | 10 ps | 35 ps | 200 ps | 5 ps |

| Imm Gen | PC (read) | Adder | PC (write) | Mem (D) (write) | Control |
|---|---|---|---|---|---|
| 45 ps | 20 ps | 155 ps | 10 ps | 15 ps | 60 ps |

b)

LW is the highest latency in existing data path, now with an extra adder in the critical path of LW to implement the Load-Add instruction, we have essentially introduced a new highest latency instruction therefore we have to increase the CT to accommodate this new instruction (always design for the worst case). So new CT is CT+155 ps. Now It is also mentioned that CT is LW's latency (existing highest latency instruction). Calculating latency of the LW instruction:



c) Rewrite the assembly code with the new "load-add" instruction. What is the improvement or slowdown in the performance of your assembly in the new hardware? (hint: think about what the clock cycle will be now. Be careful about instruction count too!).

```
New assembly with Load-add:

           ADDI x2, x0, 0
           ADDI x5, x0, 0
           ADDI x7, x0, 1000
LOOP:      LWADD x5, x2, x5
           ADDI x2, x2, 4
           BNE x2, x7, LOOP
           SW x2, x5, 4
```

With the implementation of this new instruction we have decreased the Instruction Count and increased the Cycle Time both of which will have an effect on the CPU time.

The loop runs 250 times on 3 instructions, so we have a total IC of 3*250 instructions in the loop along with 4 instructions making the total of 3*250+4 = 754 instructions in new implementation.

And similarly the original implementation would have an IC of 4*250+4 = 1004

Q2 [50 points] Consider the following instruction sequence:

```
begin:
     add x1, x0, 1
     beq  x1, x0, end
     addi x2, x1, 1
     sw x2, x1, 3
     addi x3, x2, 7
     lw x3, x0, 4
     or x4, x3, x1
end:
     xor x4, x4, x0
```

Answer the following questions.

a. Assume that we have a 5-stage pipeline processor with forwarding (i.e., we can forward either from MEM to EX or WB to EX). Assume that the branches are resolved in the MEM stage, and we always predict NOT TAKEN. Draw a table and show the pipeline timeline for each instruction (by putting f,d,e,m,w for each instruction). For stall, either repeat the stage or use (-). If there is a forwarding, show it with * (for both the forwarder and forwardee).

a)

```
begin:
     add x1, x0, 1
     beq  x1, x0, end      RAW in x1
     addi x2, x1, 1        RAW in x1
     sw x2, x1, 3          RAW in x2
     addi x3, x2, 7        RAW in x2
     lw x3, x0, 4
     or x4, x3, x1         LoadUse in x3
end:
     xor x4, x4, x0        RAW in x4
```

Assumptions:
5 stage pipelined processor, With forwading
Branches are resolved at mem stage
Branch always not taken, and prediction of Not taken was accurate, when resolved in mem stage.
***->Forwader and *->Forwadee**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| add | f | d | e | ***m** | ***w** | | | | | | | | |
| beq | | f | d | ***e** | m | w | | | | | | | |
| addi | | | f | d | ***e** | ***m** | ***w** | | | | | | |
| sw | | | | f | d | ***e** | m | w | | | | | |
| addi | | | | | f | d | ***e** | m | w | | | | |
| lw | | | | | | f | d | e | m | ***w** | | | |
| or | | | | | | | f | d | **-** | ***e** | ***m** | w | |
| xor | | | | | | | | f | - | d | ***e** | m | w |

b. Now assume that we can only forward from the WB stage (i.e., we have to stall if forwarding is needed from the MEM stage). What is the new CPU Time for this processor?

New assumptions: Can only forward from WB stage, so need to stall the first RAW affected instruction for 1 cycle, the second RAW affected instruction and LoadUse remain the same.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| add | f | d | e | m | ***w** | | | | | | | | | | | |
| beq | | f | d | **-** | ***e** | m | w | | | | | | | | | |
| addi | | | f | d | **-** | e | m | ***w** | | | | | | | | |
| sw | | | | f | - | d | **-** | ***e** | m | w | | | | | | |
| addi | | | | | | f | **-** | d | e | m | w | | | | | |
| lw | | | | | | | | f | d | e | m | ***w** | | | | |
| or | | | | | | | | | f | d | **-** | ***e** | m | ***w** | | |
| xor | | | | | | | | | | f | - | d | **-** | ***e** | m | w |

```
CPU time = IC * CPI * CT
         = 16*CT (as IC*CPI =  no of cycles needed)
Assuming CT is same
```

Q3 [3 points] What is the first item on your bucket list? {The craziest might get an additional bonus :D — Ask us if you want to know ours ;)}

```
Umm, Learn a new language (japenese? French? German?) nay,
decide which new language to learn! :P
```