LSU EE 4720 Homework 4 Solution Due: 1 April 2020

It's up to all of us: $r > 2\,\mathrm{m} \Rightarrow R_e < 1$ where r is the radius of the largest circle with you at the center and containing only people in your household, and R_e is the effective reproduction number, the number of people infected by an infected person.

Problem 1: Appearing below is the code fragment from Homework 3.

```
# Cycle         0     1     2     3     4     5     6
addi R2, r0, 0  IF    ID    EX    ME    WB
mul R1, R2, r3        IF    ID    EX    ME    WB
add r4, R2, R1              IF    ID    EX    ME    WB
# Cycle         0     1     2     3     4     5     6
```

(*a*) Does this code fragment look like it was compiled with optimization on?

If your answer is something like "yes, it could be part of optimized code" then explain why you think it so and provide any missing context. (Do not change or re-arrange the three instructions above.)

If your answer is something like "no, it does not appear optimized" then show what the code would look like after optimization. *Hint: A correct answer can start with either "Yes it does" or "No it doesn't". The "No" answer is straightforward.*

No, because with constant propagation and folding none of the instructions are necessary. We know that `r2` will be assigned 0, and then `r1` and `r4` will also be zero. Any uses of those registers in the same basic block can be replaced by zero. (See previous answer.)

Problem 2: MIPS does not appear to have a `muli` instruction.

(*a*) Comment on the following:

> *MIPS has a* `mul` *instruction but does not have a* `muli` *instruction because, as the solution to Homework 2 shows, the additional hardware for* `muli` *(beyond that used for* `mul`*) would be too costly.*

Is the statement above reasonable or unreasonable? Explain.

Unreasonable, because the connection to the lower input of the `Y1` unit is from the lower ALU mux, which has a connection to the immediate. Therefore, the implementation of the `muli` would only require changes to control logic.

(*b*) Show the encoding of MIPS instruction `mul r1, r2, r3`. Show all 32 bits of the instruction, divided into fields (each field can be shown in the radix of your choice). (The MIPS ISA manuals are linked to the course Web page. Instruction encodings are in Volume II.)

A quick lookup in the ISA manual reveals that the opcode is $1c_{16}$ and the Func field value is 2. As with most type-R instructions the order of the assembly language arguments are `rd`, `rs`, `rt`. The `sa` field is—must be—zero.

	Opcode	rs	rt	rd	sa	Func
MIPS R:	0x1c	2	3	1	0	0x2
	31 26	25 21	20 16	15 11	10 6	4 0

(*c*) Some possible reasons that there is no `muli` instruction in MIPS is that either there are no Format-I opcodes available (they are all used by other instructions) or that the few remaining opcodes are being kept in reserve for a better instruction than a `muli`.

Based on the MIPS Architecture Manuals (they are linked to the course references page) how many opcodes are available for new Format-I instructions? The easy way to solve this is to find the right table. The hard way to solve this is to go through the 144 or so pages of instruction descriptions. *Hint: Look in volume I.*

The following solution is based on the MIPS manuals linked to the course Web page, which are Revision 0.95 and describe MIPS before Release 6.

Table A-2 shows the encoding of the Opcode field. Twelve entries in the table appear blank. Assuming they are supposed to be blank, and are not a problem with either the PDF encoding of the manual or of the PDF viewer I am using, then there are 12 opcode slots available.

Assuming there is a PDF problem of some kind then the number of free opcodes are the number of entries in Table A-2 with the symbol shown in the first row of Table A-1, the row for "Operations . . . reserved for future use" instructions.

Problem 3: Perhaps you saw this coming: Time to add `muli` to MIPS.

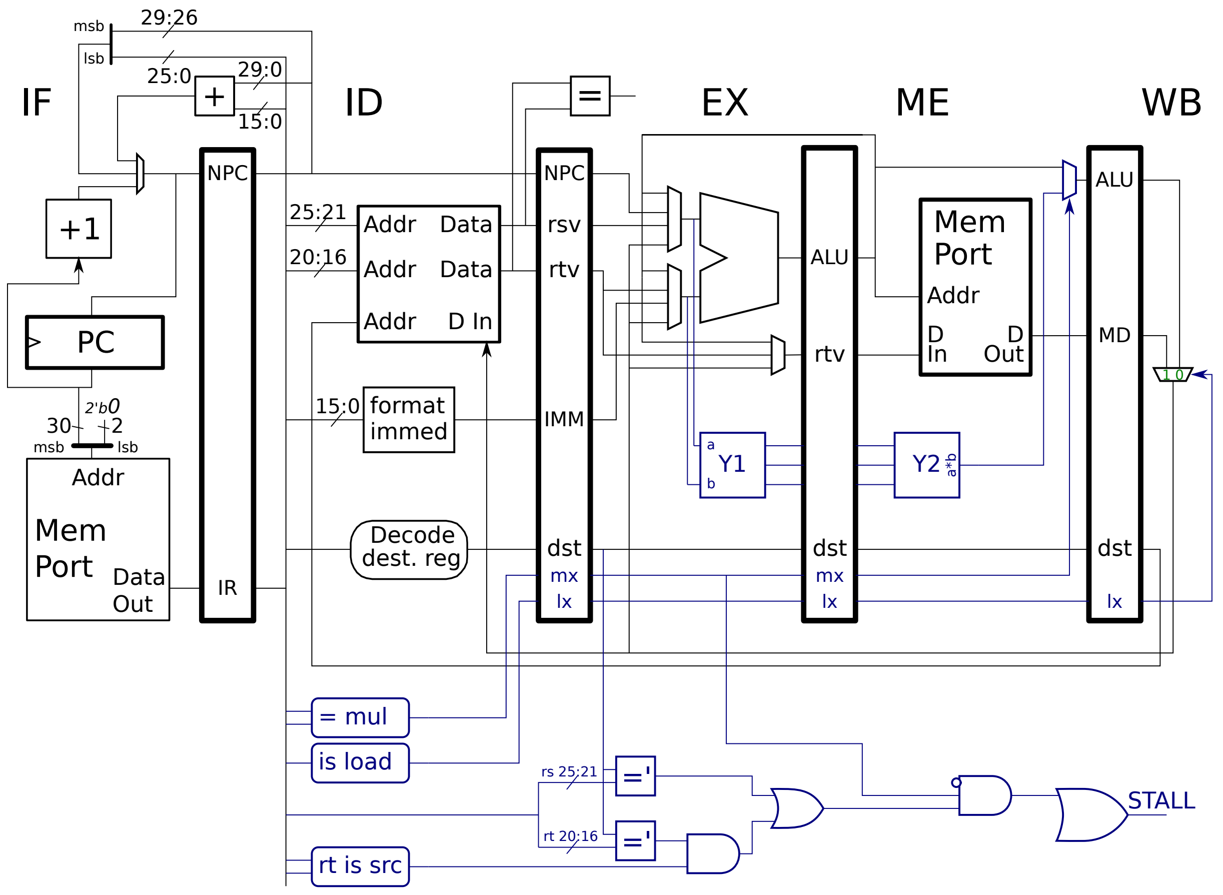
(*a*) Show how a Format-R `muli` instruction with a ten-bit immediate might be defined using unused fields in the Format-R encoding. Make up your own function field value, but try to pick one that's unused. (See the previous problem.) Show how `muli r1, r2, 43` might be encoded for your `muli` definition.

Table A-5 of the Revision 0.95 shows function field values used by the family of instructions that includes `mul`. The `mul` instruction is in the first row of the table and the second row appears empty (see the gripe about the PDF manual from previous problem's solution). Lets reserve the second row for immediate-value versions of first-row instructions. So the function field value for `mul` is $000\,010_2$. So lets make the function field for `muli` $001\,010_2$. The opcode will remain $1c_{16}$. To encode the 10-bit immediate use bits 20:16 (the `rt` field) for the upper 5 bits and use bits 10:6 (the `sa` field) for the lower 5 bits. So to encode $43_{10} = 00\,0010\,1011_2$ set the `rt` field to 00001_2 and the `sa` field to $01011_2 = b_{16}$.

	Opcode	rs	rt	rd	sa	Func
MIPS R:	0x1c	2	1	1	0xb	0x2
	31 26	25 21	20 16	15 11	10 6	4 0

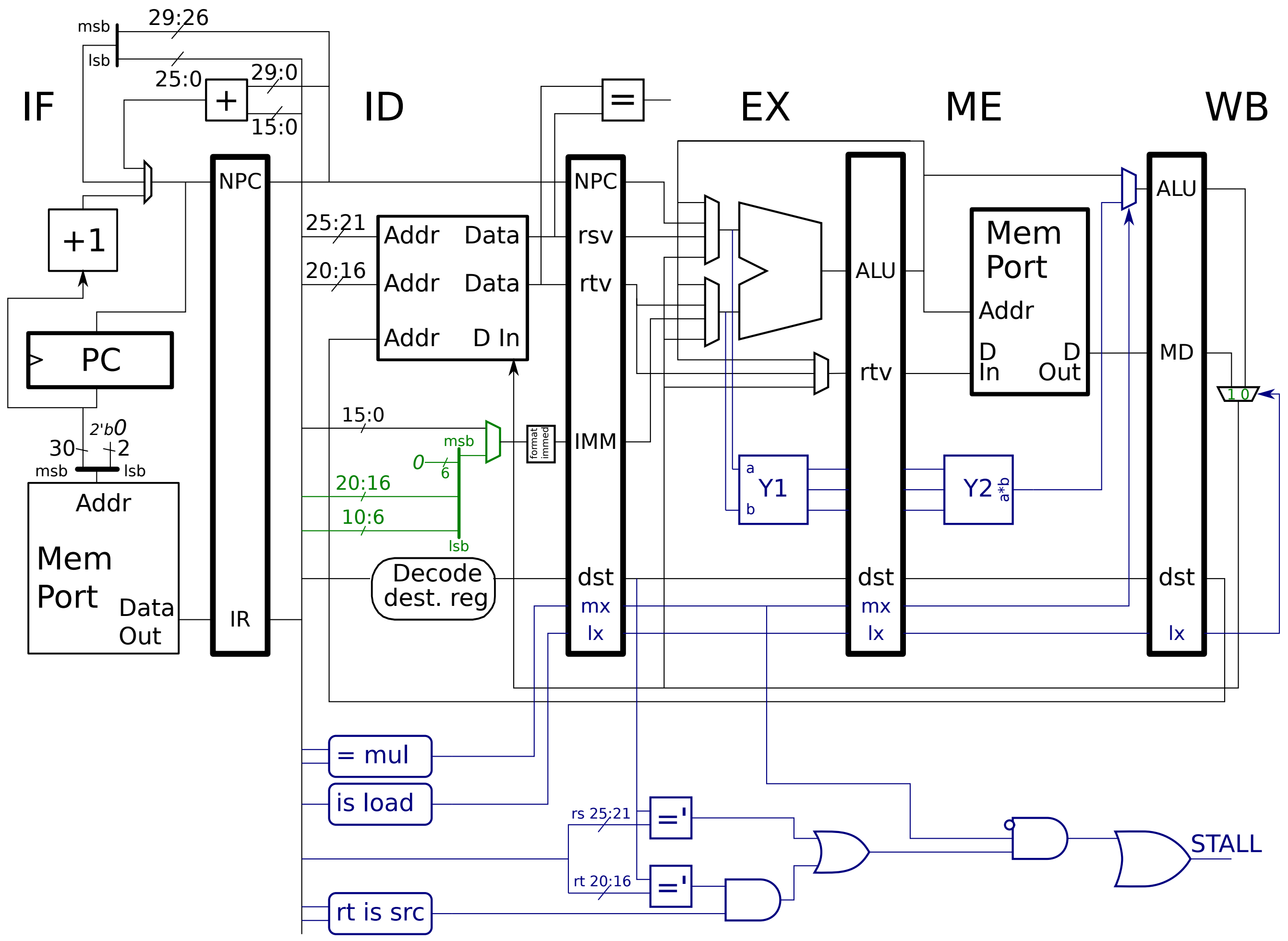
(*b*) Modify the hardware below (there's a copy on the next page) to implement this new instruction. The modified hardware should provide the immediate needed by `muli`. Show datapath **but not** control logic. Of course, any changes should not break existing instructions.

Pay attention to cost and performance. This can easily be solved by adding a mux in the `ID` stage. *Hint: The solution is not much more than a mux. Be sure to carefully label the inputs.*



The SVG source for the diagram below is available at
`https://www.ece.lsu.edu/ee4720/2020/hw03-p2.svg`.

Solution appears below in green. Since no other instruction concatenates the `rt` and `sa` fields to extract an immediate, that hardware had to be added. The lower input to the new ID-stage mux consists of those concatenated fields and also includes 6 zero bits on the most-significant side. The resulting constant is 16 bits, the same size as ordinary immediate. The format immed block sign-extends the value to 32 bits, which will have no effect for the `muli` immediate but is still needed for the other immediates.



IF
ID
EX
ME
WB
msb
lsb
29:26
25:0
29:0
15:0
+
NPC
+1
PC
2'b0
30
2
msb
lsb
Addr
Mem Port
Data Out
IR
25:21
20:16
Addr Data
Addr Data
Addr D In
15:0
msb
0
6
20:16
10:6
lsb
format immed
Decode dest. reg
=
NPC
rsv
rtv
IMM
dst
mx
lx
ALU
rtv
dst
mx
lx
Mem Port
Addr
D In
D Out
ALU
MD
dst
lx
1 0
a
b
Y1
Y2
a*b
= mul
is load
rt is src
rs 25:21
rt 20:16
='
='
STALL