EE309 Bonus Project
Pipelined IITB RISC 22

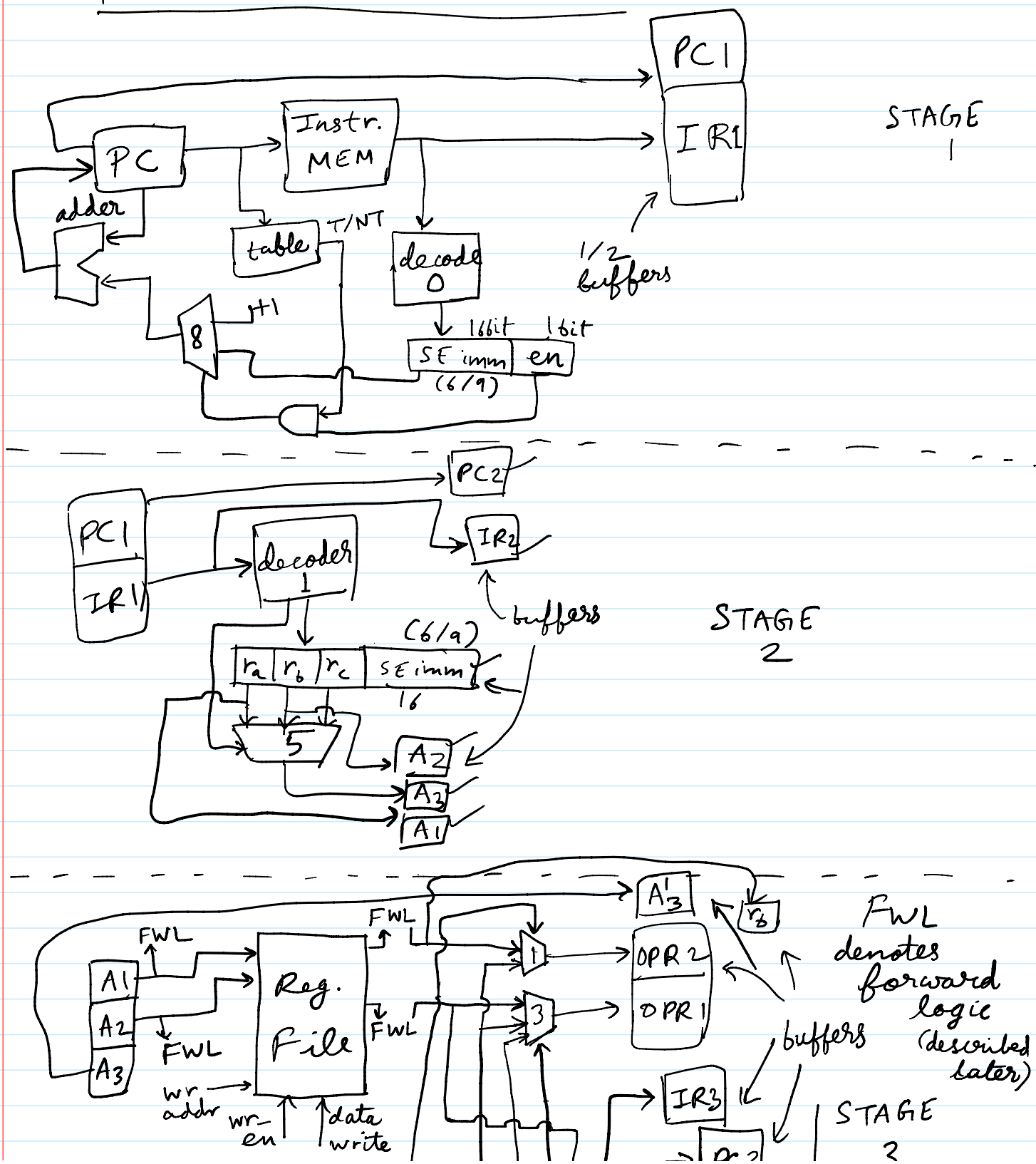
Team Members (Name, Roll no):

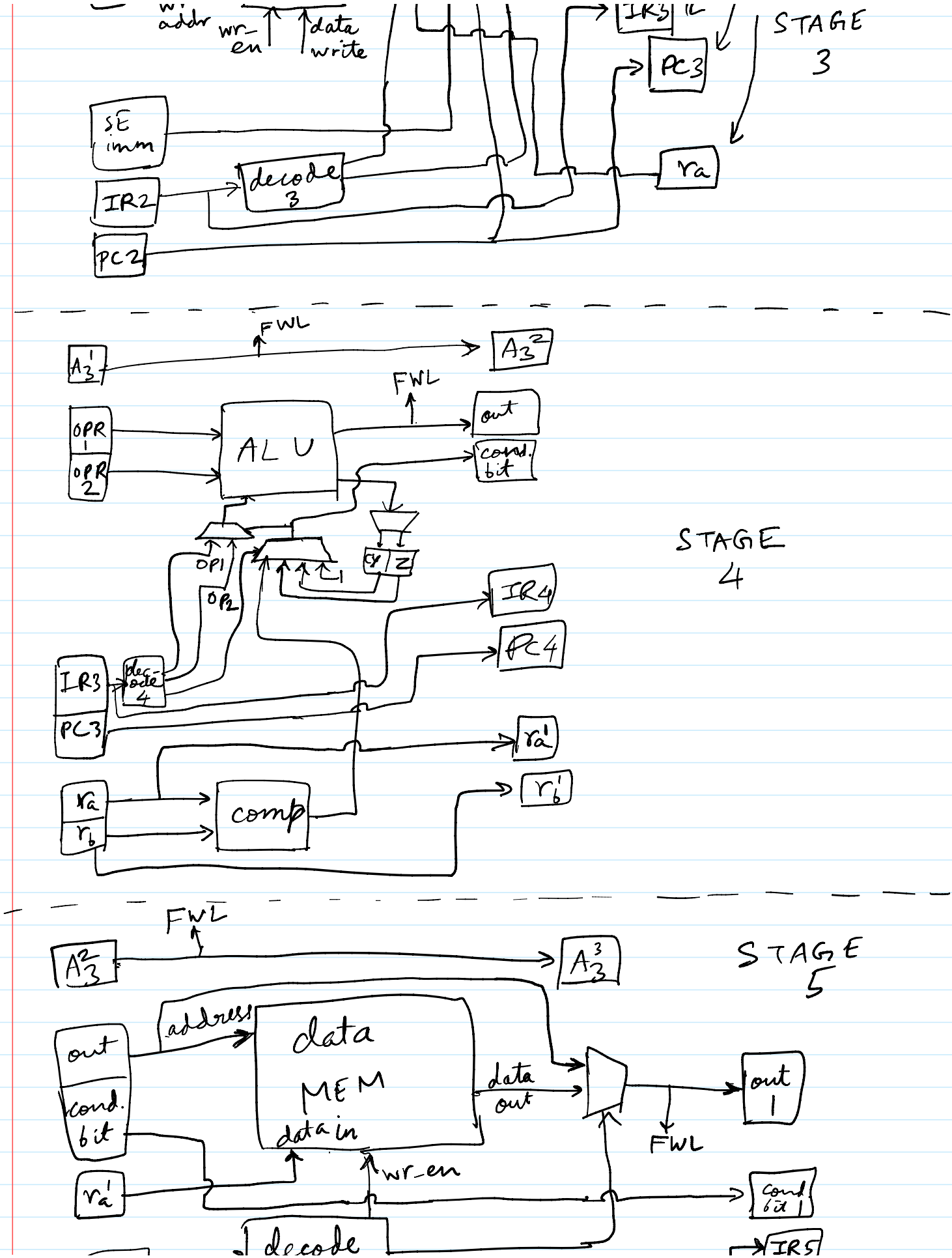
1) Durgaprasad Prakash Bhat, 200070017
2) Waqar Mirza, 200070090
3) Kalpit Borkar, 200070029
4) Dadhichi Telwadkar Dattatraya, 20d070083

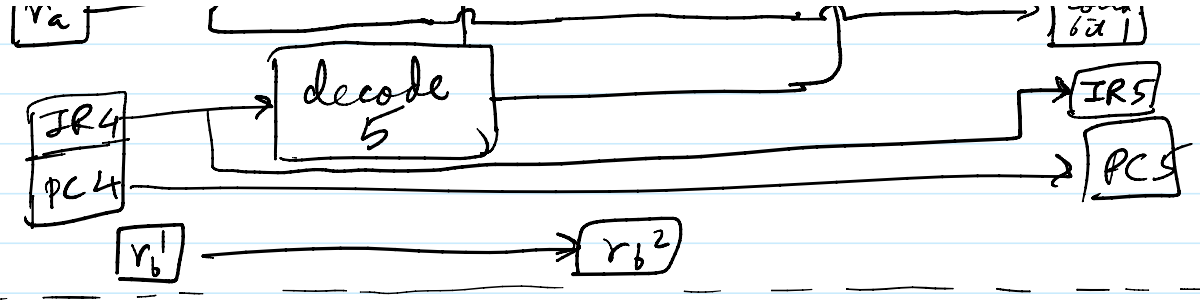
DATAPATH

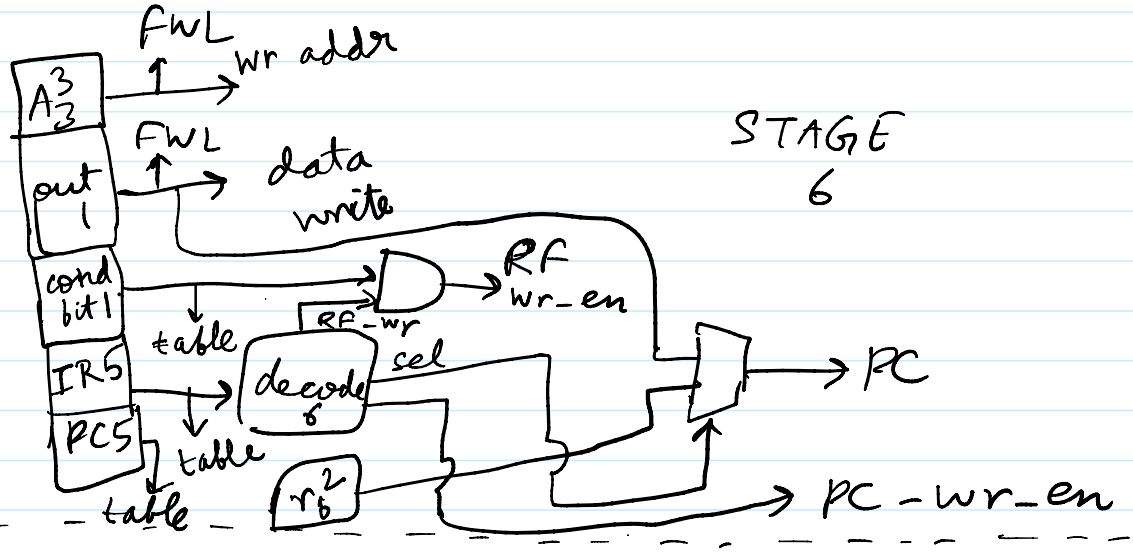
07 May 2022 16:00

FINAL DESIGN




STAGE 3
addr
wr_en
data write
IR3
PC3
SE imm
decode 3
IR2
PC2
ra
FWL
A3^1
A3^2
OPR 1
OPR 2
ALU
out
cond. bit
OP1
OP2
CY
Z
IR4
PC4
IR3
decode 4
PC3
ra'
rb'
ra
rb
comp
STAGE 4
A3^2
A3^3
STAGE 5
address
data MEM
data in
data out
out 1
cond. bit
ra'
wr_en
cond bit
decode
IR5

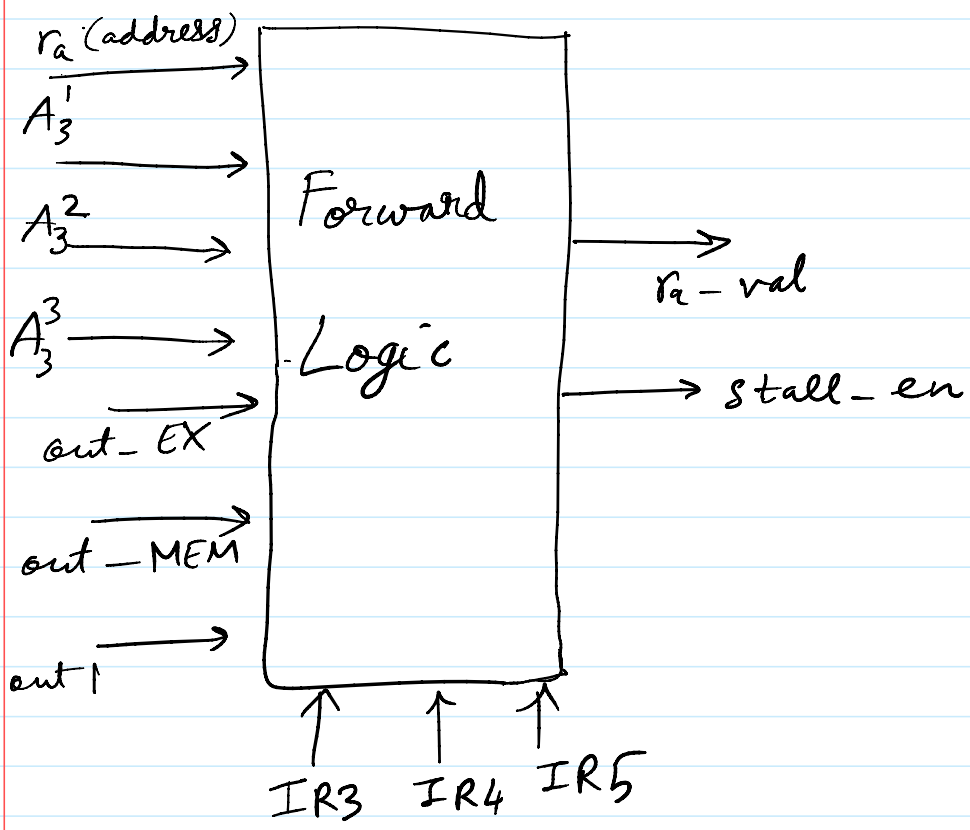




STAGE 6

Forward Logic:

For r_a: (same logic for r_b with r_a replaced with r_b)



Logic: see which instruction modifies r_a most recently, and access output from closest stage when it is available and assign updated output to r_a - val.

If output for the latest instruction is available only after MEM stage (out_MEM, this only happens in LW) then we need to stall pipeline till LW instr. is in MEM stage.

Decoder designs:

decode 0:

if instr == JAL

output ⇒ SE imm(9) & 1_{en1} & 1_{en2}

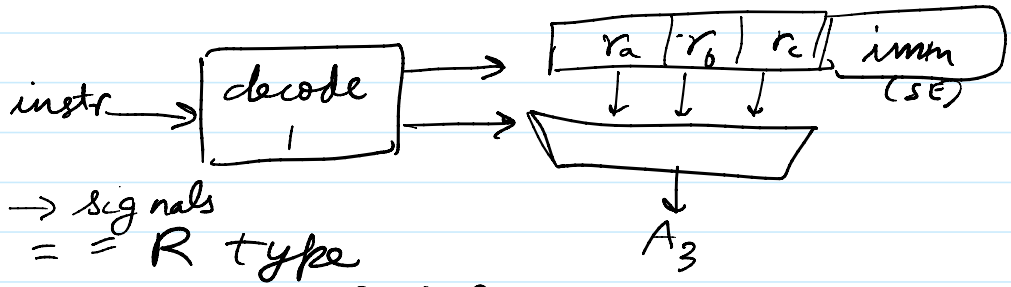
if instr == BEQ

output ⇒ SE imm(6) & 1_{en1} & 0_{en2}

else

output ⇒ 0

Decode 1:



r_a, r_b, r_c → signals

if instr == R type

$r_a \Rightarrow r_a$, $r_b \Rightarrow r_b$, $r_c \Rightarrow r_c$

imm (SE) ⇒ 0

if instr == I type

$r_a \Rightarrow r_a$, $r_b \Rightarrow r_b$, $r_c \Rightarrow 0$

imm (SE) ⇒ SE imm(6)

$r_a \Rightarrow r_a$, $r_b \Rightarrow r_b$, $r_c \Rightarrow 0$

$\underset{(SE)}{imm} \Rightarrow SE\ imm(6)$

if instr == J type

$r_a \Rightarrow r_a$, $r_b, r_c \Rightarrow 0$

$\underset{(SE)}{imm} \Rightarrow SE\ imm(9)$

if ADD, ADZ, ADC, ADL, NDU, NDC, NDZ

$A_3 \Rightarrow r_c$

if ADI

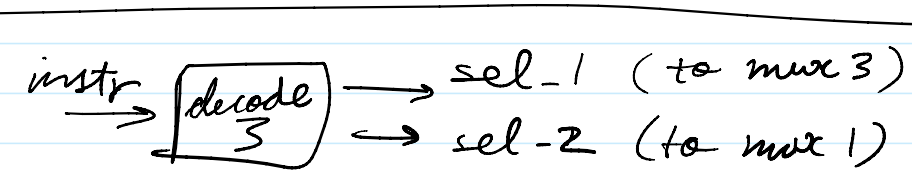
$A_3 \Rightarrow r_b$

if rest

$A_3 \Rightarrow r_a$

output $\Rightarrow$ $r_a, r_b, A_3, \underset{(SE)}{imm}$

<u>decode 3:</u>



if ADD, ADZ, ADC, ADL, NDU, NDC, NDZ

select: mux 3 : r_a

mux 1 : r_b

if ADI, JRI

mux 3 : r_a

mux 1 : Imm

if LW, SW, LHI

mux 3: Imm

mux 1: r_b

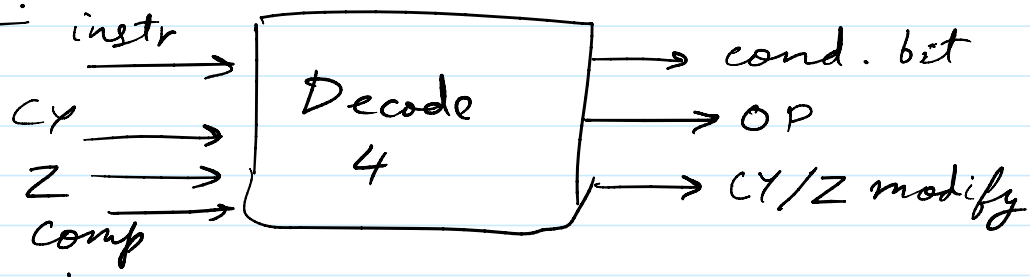
if BEQ, JAL, JLR

mux 3: PC

mux 1 : Imm

mux 3 : PC
mux 1 : Imm

Decode 4:



cond. bit ⇒ 1

if ADD, ADC, ADZ, ADI, LW, SW, JRI

- OP ⇒ add
- if (ADC && CY==0) || (ADZ && Z==0)
  - cond. bit ⇒ 0, CY/Z modify ⇒ 0
- else if ADD, ADC, ADZ, ADI
  - CY/Z modify ⇒ 1
- else
  - CY/Z modify ⇒ 0

if ADL

- OP ⇒ SHIFT-ADD (adds after shifting OPR-2 by 1 bit)
- CY/Z modify ⇒ 1

if NDU, NDZ, NDC

- OP ⇒ nand
- if (NDC && CY==0) || (NDZ && Z==0)
  - cond. bit ⇒ 0, CY/Z modify ⇒ 0
- else
  - CY/Z modify ⇒ 1

if LHI

- OP ⇒ shift 7 (shifts OPR1 by 7 bits)
- CY/Z modify ⇒ 0

if BEQ

- CY/Z modify ⇒ 0

if

CY/Z modify ⇒ 0
if comp == equal
OP ⇒ add
else
OP ⇒ increm. , cond. bit ⇒ 0
↳ (increments OPR1)

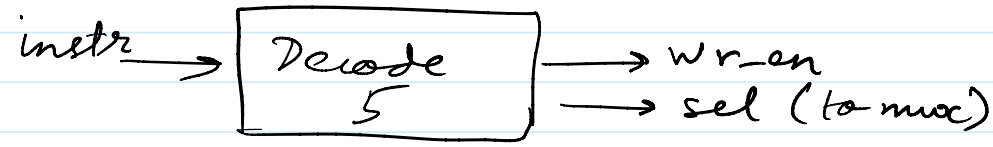
if JAL, JLR

OP ⇒ increm.
CY/Z modify ⇒ 0

DecodersPg2

09 May 2022 14:12

Decoder 5:



wr_en ⇒ 0

sel ⇒ out (meaning mux selects out)

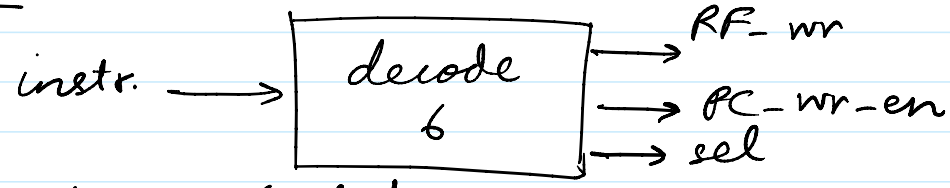
if SW

wr_en ⇒ 1

if LW

sel ⇒ data out (mux selects data out)

Decode 6:



sel ⇒ out1 → (selects out1)

PC_wr_en ⇒ 0

RF_wr ⇒ 1

if SW, BEQ, JRI

RF_wr ⇒ 0

if BEQ, JLR, J RI

PC-wr-en ⇒ 1 (for BEQ some additional conditions need to be met)

if JLR

sel ⇒ r_b^2 (selects r_b^2)

If prediction != cond. bit1

PC-wr-en ⇒ 1

els if prediction == cond. bit1

PC-wr-en ⇒ 0

To fetch prediction

fetch — cond. br.

prediction

from table

send PC5 into table

PC - wr - en => 0

Decoders control everything except stalling pipeline.

↳ for stalling we need a separate control

Pipeline stalling for dependency hazard mitigation and branching

JAL is already handled in stage1
For JLR and JRI

B1	B2	B3	B4	B5	
1	1	1	1	1	cycle 1
0	1	1	1	1	cycle 2
0	0	1	1	1	cycle 3
0	0	0	1	1	cycle 4
0	0	0	0	1	cycle 5
1	0	0	0	0	cycle 6
1	1	0	0	0	cycle 7
1	1	1	0	0	cycle 8
1	1	1	1	0	cycle 9
1	1	1	1	1	cycle 10

Failed prediction in BEQ

B1	B2	B3	B4	B5	
0	0	0	1	1	cycle 1 (the BEQ is in the EX stage at this point)
0	0	0	0	1	cycle 2
1	0	0	0	0	cycle 3
1	1	0	0	0	cycle 4
1	1	1	0	0	cycle 5
1	1	1	1	0	cycle 6
1	1	1	1	1	cycle 7

The case when an instruction uses a register which is also being loaded from memory in the previous instruction

B1	B2	B3	B4	B5	
0	0	0	1	1	cycle 1
1	1	1	1	1	cycle 2

After cycle 2 the memory output is available and that is provided as operands to the Ex stage via the forwarding logic.

Here B1, B2, B3, B4 and B5 denote the write enable of the buffers at the interface of the stages. Hence, B1= ‘1’ means all the buffers at the interface between stage1 and stage2 are enabled to be written at the end of the cycle.