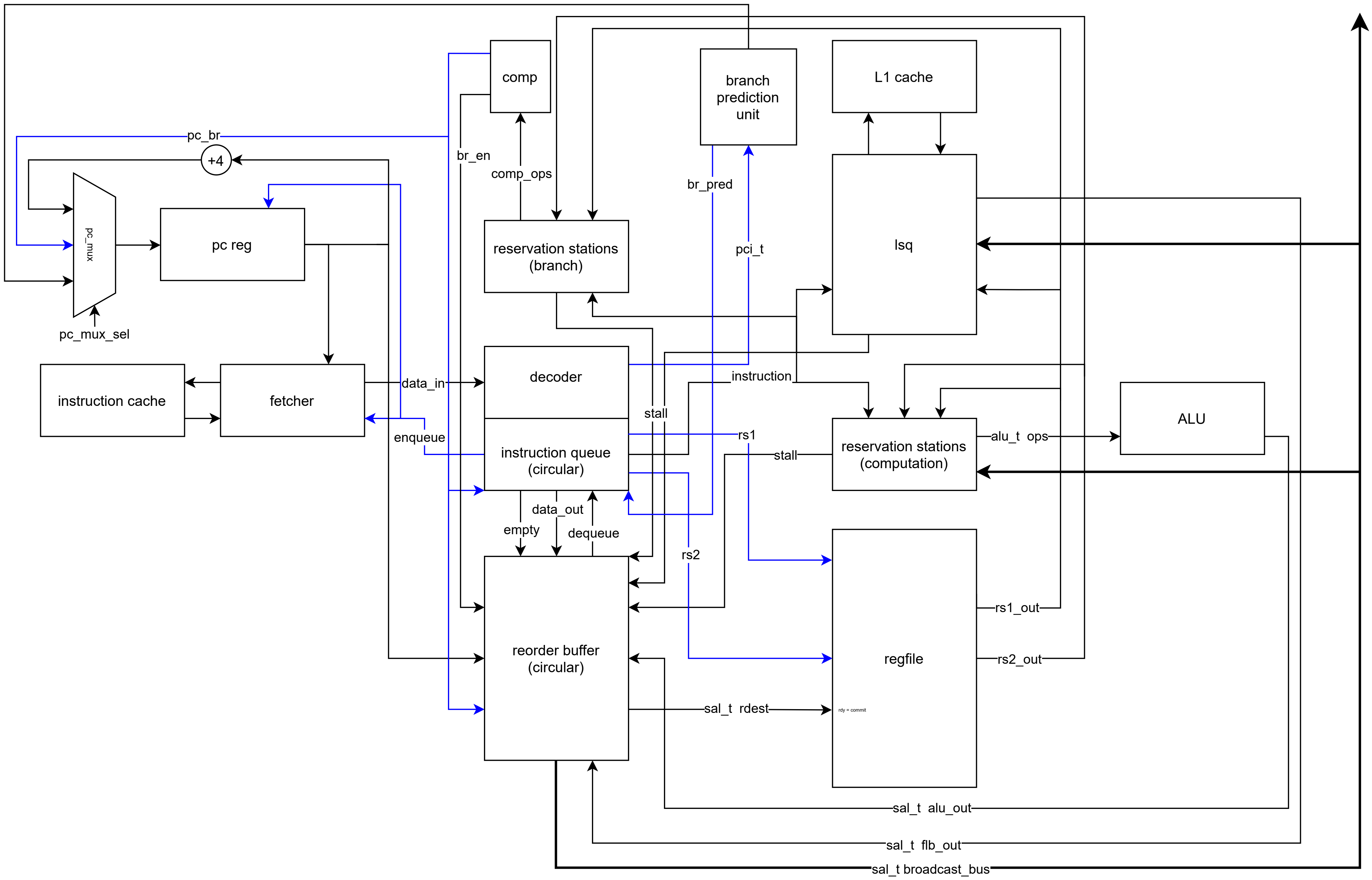
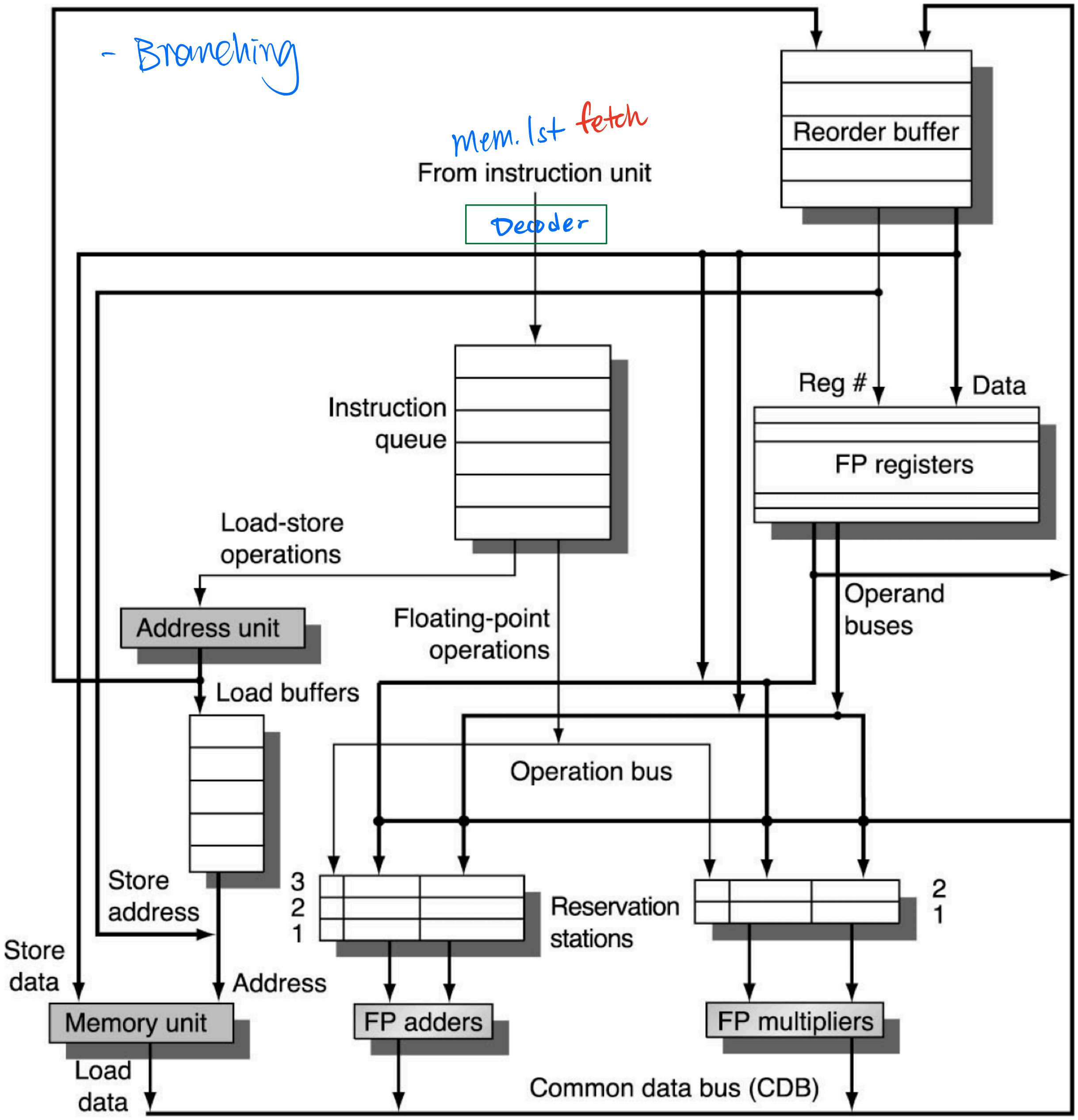

pc_br
+4
pc_mux
pc reg
pc_mux_sel
instruction cache
fetcher
data_in
enqueue
comp
br_en
comp_ops
reservation stations
(branch)
branch
prediction
unit
br_pred
pci_t
L1 cache
lsq
decoder
instruction queue
(circular)
instruction
stall
rs1
stall
rs2
reservation stations
(computation)
alu_t ops
ALU
data_out
empty
dequeue
reorder buffer
(circular)
regfile
rs1_out
rs2_out
sal_t rdest
rdy = commit
sal_t alu_out
sal_t flb_out
sal_t broadcast_bus

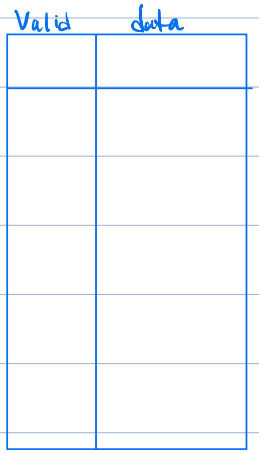

- Branching
mem. 1st fetch
From instruction unit
Decoder
Reorder buffer
Reg #
Data
FP registers
Instruction queue
Load-store operations
Address unit
Floating-point operations
Operand buses
Load buffers
Operation bus
Store address
3
2
1
Reservation stations
2
1
Store data
Address
Memory unit
FP adders
FP multipliers
Load data
Common data bus (CDB)

extras ⓣ

- l2 cache    - ROB multiple commits
- branch pred    - ROB multiple broadcasts

instr
## Circular Queue : Circular Queue

| input: | out | internal |
|---|---|---|
| clk | dataout | front |
| rst, | full | end |
| datain | | size |
| enqueue | | |
| dequeue | | |

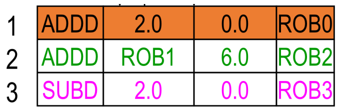


## Reservation Stations

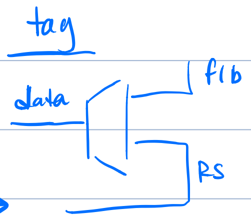
not circular queue

need to track vacancy

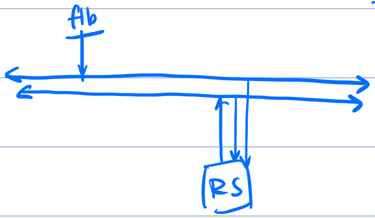
adders:

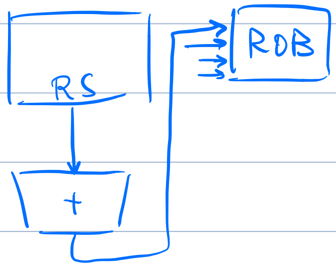


| | | | | |
|---|---|---|---|---|
| 1 | ADDD | 2.0 | 0.0 | ROB0 |
| 2 | ADDD | ROB1 | 6.0 | ROB2 |
| 3 | SUBD | 2.0 | 0.0 | ROB3 |



| inputs | outputs |
|---|---|
| sal_t_i (R0) | alu_t_o x 8 |
| sal_t_i (R1) | |
| operation_i | |
| sal_t_i (rob) | |





sal_t

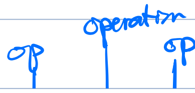
| | | |
|---|---|---|
| tag_brd_rs<br>rdy<br>data_brd_rs | tag_brd_rs<br>rdy<br>data_brd_rs | tag_brd_rs<br>rdy<br>data_brd_rs |

alu_t

| |
|---|
| sal_t |
| sal_t |
| operation |
| tag |

pci_t

| |
|---|
| pc |
| instruction |
| is_br_inst |
| br_pred |



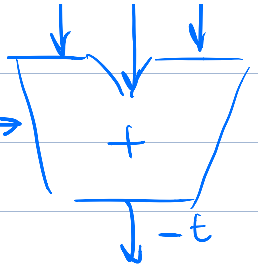
## ALU ×8

inputs

alu-t_i

outputs

sal_t_o



## flb (load)

inputs

mem_resp

mem_rdata_i

instr_Queue
reg-file
rob
memory

pci_t_i

sal_t_i (reg-file)

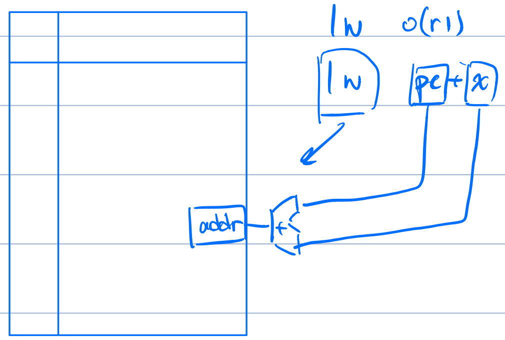
Ⓣ sal_t_i (rob)

outputs

sal_t_o (rob)

mem_read_o

flb: Circular Queue



Maybe merge?

## fsb (store)

inputs

sal_t_i (reg_f)

sal_t_i (rob)

pci_t_i (instr_q)

mem_resp (memory)

outputs

mem_write

mem_wdata

## regfile

inputs

sal_t_i (rob)

don't use tag

reg_num_i (rob)

outputs

sal_t_o (reg_bus) x2

flb fsb rs

| | | |
|---|---|---|
| F0 | 0.0 | |
| F2 | 2.0 | |
| F4 | 4.0 | ROB0 |
| F6 | 6.0 | ROB2 |
| F8 | 8.0 | ROB1 |

## ROB :: circular Queue

dequeue to inst queue if low = full

inputs

reg_num_i (inst queue)

~~enqueue_i~~ (inst queue)

outputs

full_o (inst queue)

⊕ sal_t_o (comm bnd bus) → broad cast whenever

reg_num_o (reg file)

⊕ sal_t_o (reg file) → only when dequeue

| | | | |
|---|---|---|---|
| 0 | MULD | F8 | - |
| 1 | MULD | F8 | - |
| 2 | ADDD | F6 | - |
| 3 | SUBD | F8 | 2.0 |
| 4 | SUBI | | |
| 5 | BNEZ | | |
| 6 | ADDD | F4 | |

store PC value in ROB

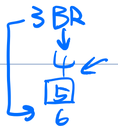
## Branch Unit

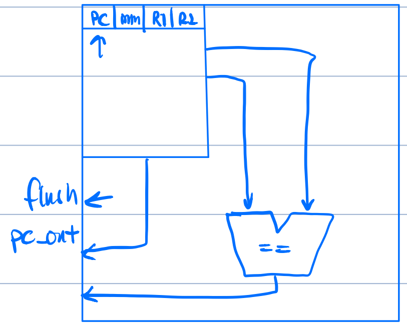
input

sal_t_i (reg file) x2

pci_t_i (inst queue)

output





## Branch Predictor

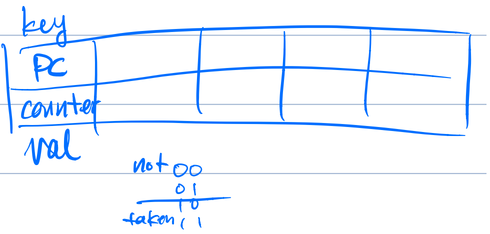
inputs

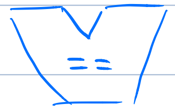
pci_t_i (inst queue)

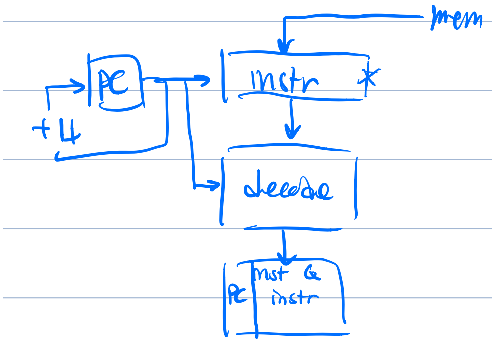
~~sal_t_i (reg file) x2~~

sal_t_i (rob)

outputs







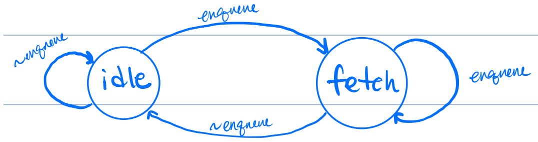
- instruction vs mem acc? → 2 caches
- adders? → one for each RS
- controller?

- most performance inc comes from load / st
- more RS → cycle time longer → slower processing

pc load = 0 when iq full

# Controls

## fetcher



## instruction queue

datacnt = head

empty = head == tail

## reservation stations

stall = 1 if full && next instr is computational

## pc logic

pc_load = enqueue ( iq )

pc_mux_sel = default = pc + 4

branch_pred if br_pred = 1

exe_br if br_en != br_pred