Lecture 10
Processor Microarchitecture (Part 1)

Xuan ‘Silvia’ Zhang
Washington University in St. Louis

http://classes.engineering.wustl.edu/ese566/

Key Concepts in Computer Architecture

- Transaction latency
  - the time to complete a single transaction
- Execution time/total latency
  - the time to complete a sequence of transactions
- Throughput
  - the number of transaction executed per unit time

Analyzing Processor Performance

$$\frac{\text{Time}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Time}}{\text{Cycles}}$$

- Instructions / program depends on source code, compiler, ISA
- Cycles / instruction (CPI) depends on ISA, microarchitecture
- Time / cycle depends upon microarchitecture and implementation

Using our first-order equation for processor performance and a functional-level model, the execution time is just the number of dynamic instructions.

Analyzing Processor Performance

$$\frac{\text{Time}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Time}}{\text{Cycles}}$$

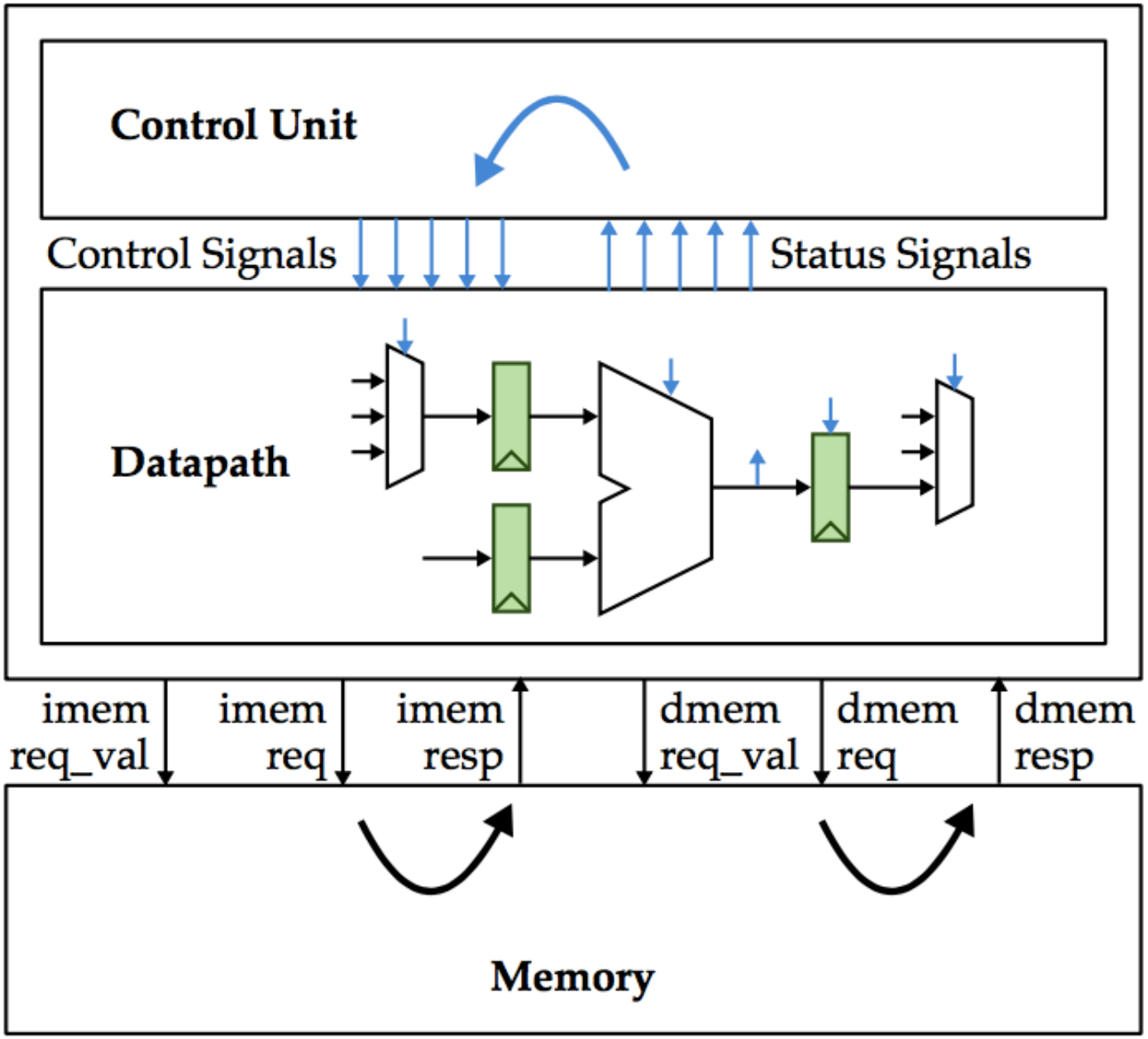
Microarchitecture	CPI	Cycle Time
Single-Cycle Processor	1	long
FSM Processor	>1	short
Pipelined Processor	≈1	short

Transactions and Steps

- Each instruction as a transaction
- Executing a transaction involves a sequence of steps

	addu	addiu	mul	lw	sw	j	jal	jr	bne
Fetch Instruction	✓	✓	✓	✓	✓	✓	✓	✓	✓
Decode Instruction	✓	✓	✓	✓	✓	✓	✓	✓	✓
Read Registers	✓	✓	✓	✓	✓			✓	✓
Register Arithmetic	✓	✓	✓	✓	✓				✓
Read Memory				✓					
Write Memory					✓				
Write Registers	✓	✓	✓	✓			✓		
Update PC	✓	✓	✓	✓	✓	✓	✓	✓	✓

Microarchitecture: Control/Datapath Split

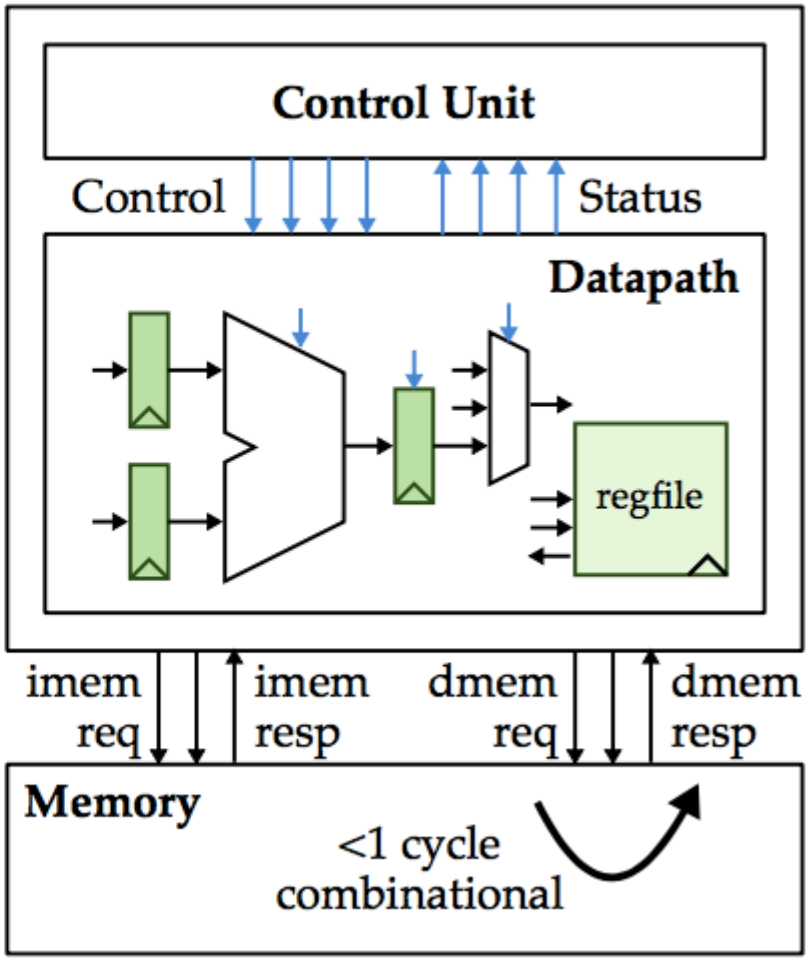


Microarchitecture	CPI	Cycle Time
Single-Cycle Processor	1	long
FSM Processor	>1	short
Pipelined Processor	≈1	short

PARCv1 Single-Cycle Processor

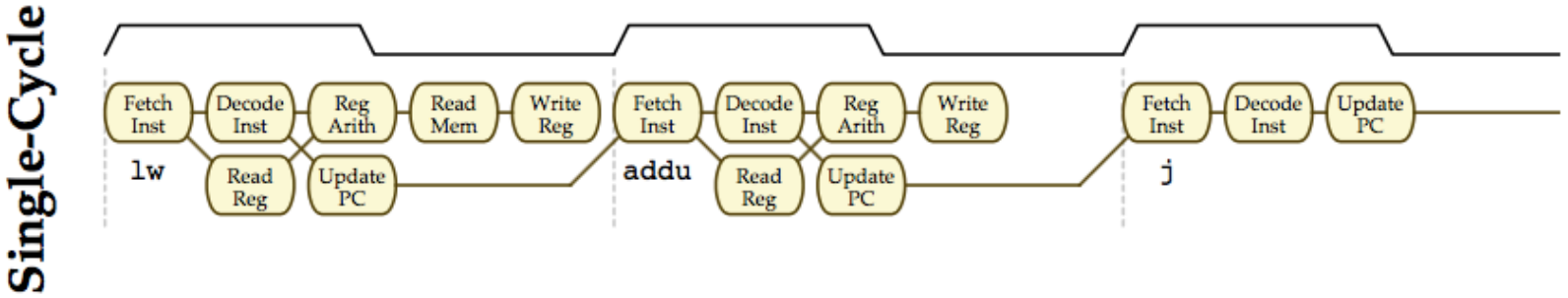
Technology Constraints

- Assume technology where logic is not too expensive, so we do not need to overly minimize the number of registers and combinational logic
- Assume multi-ported register file with a reasonable number of ports is feasible
- Assume a dual-ported combinational memory



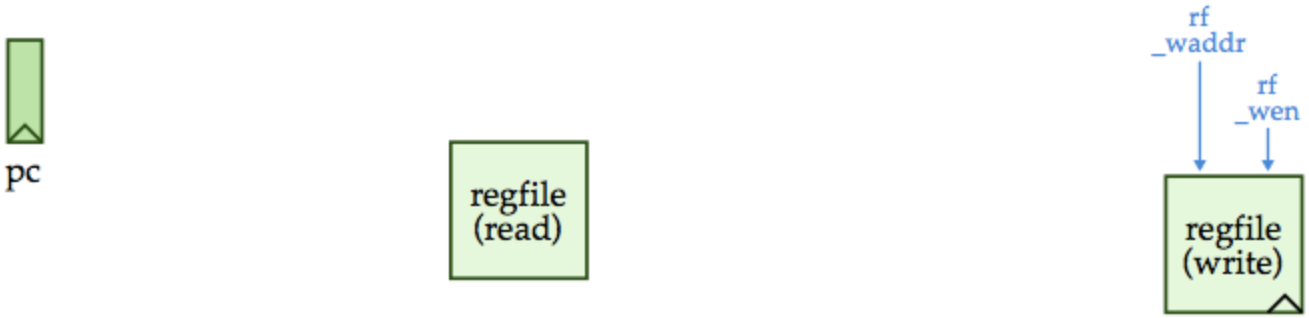
High-level Idea for Single-Cycle Processors

	addu	addiu	mul	lw	sw	j	jal	jr	bne
Fetch Instruction	✓	✓	✓	✓	✓	✓	✓	✓	✓
Decode Instruction	✓	✓	✓	✓	✓	✓	✓	✓	✓
Read Registers	✓	✓	✓	✓	✓			✓	✓
Register Arithmetic	✓	✓	✓	✓	✓				✓
Read Memory				✓					
Write Memory					✓				
Write Registers	✓	✓	✓	✓			✓		
Update PC	✓	✓	✓	✓	✓	✓	✓	✓	✓



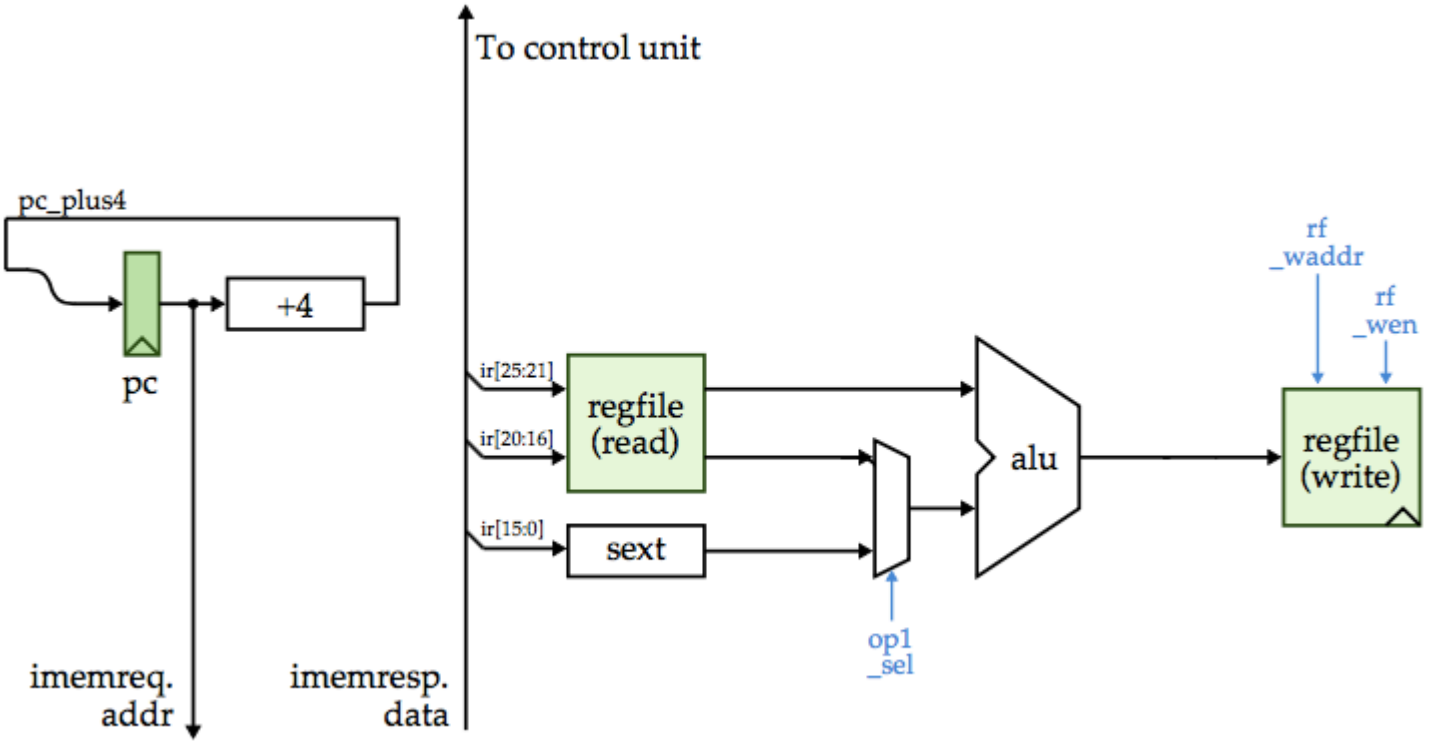
Single-Cycle Datapath

- Implement ADDU instruction
- Implement ADDIU instruction

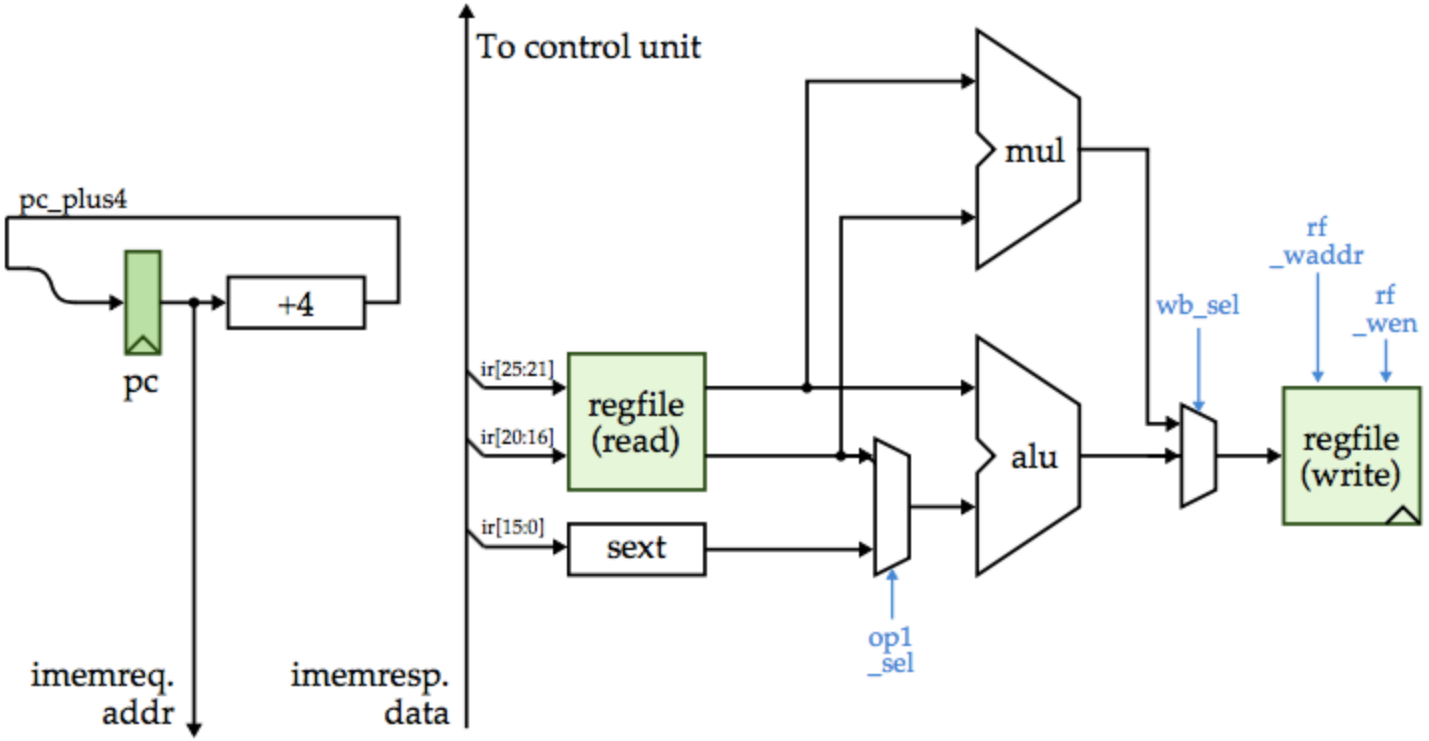


Single-Cycle Datapath

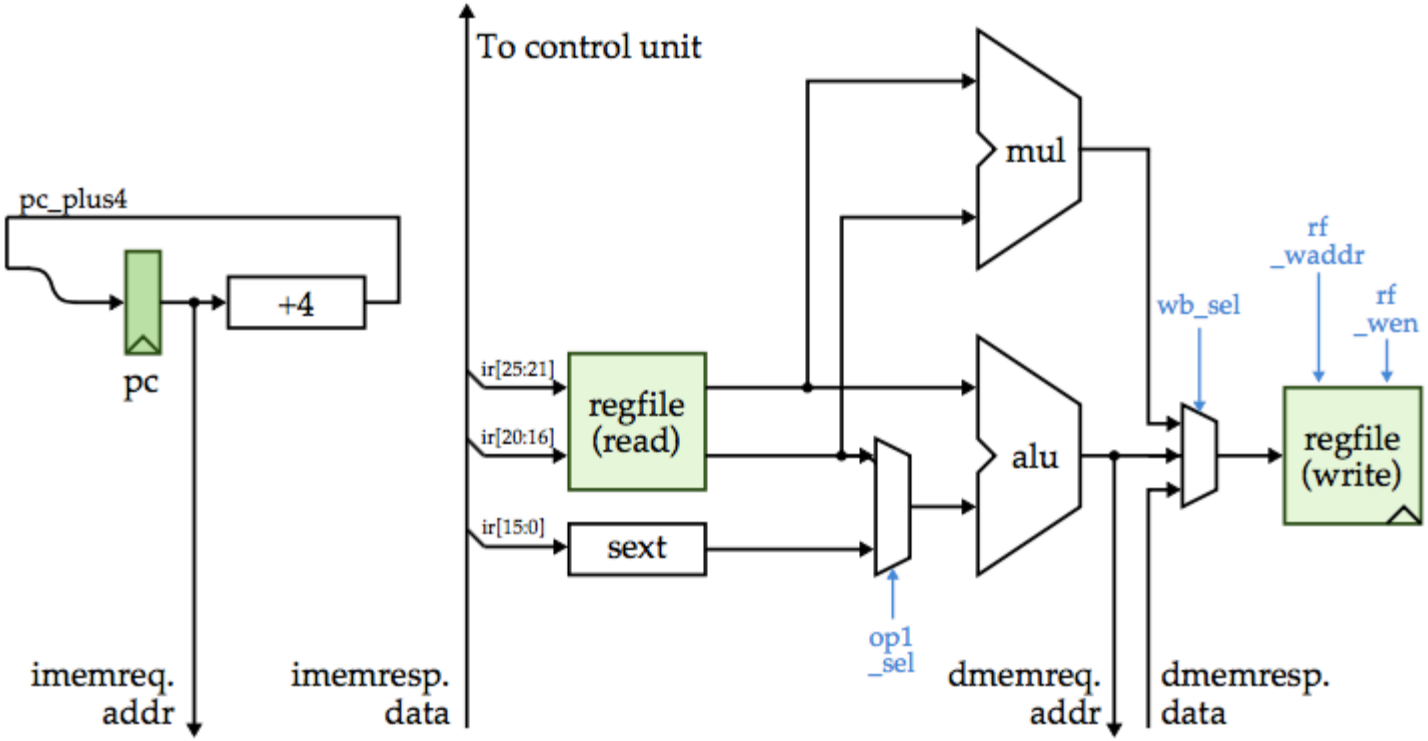
- Implement ADDU instruction
- Implement ADDIU instruction



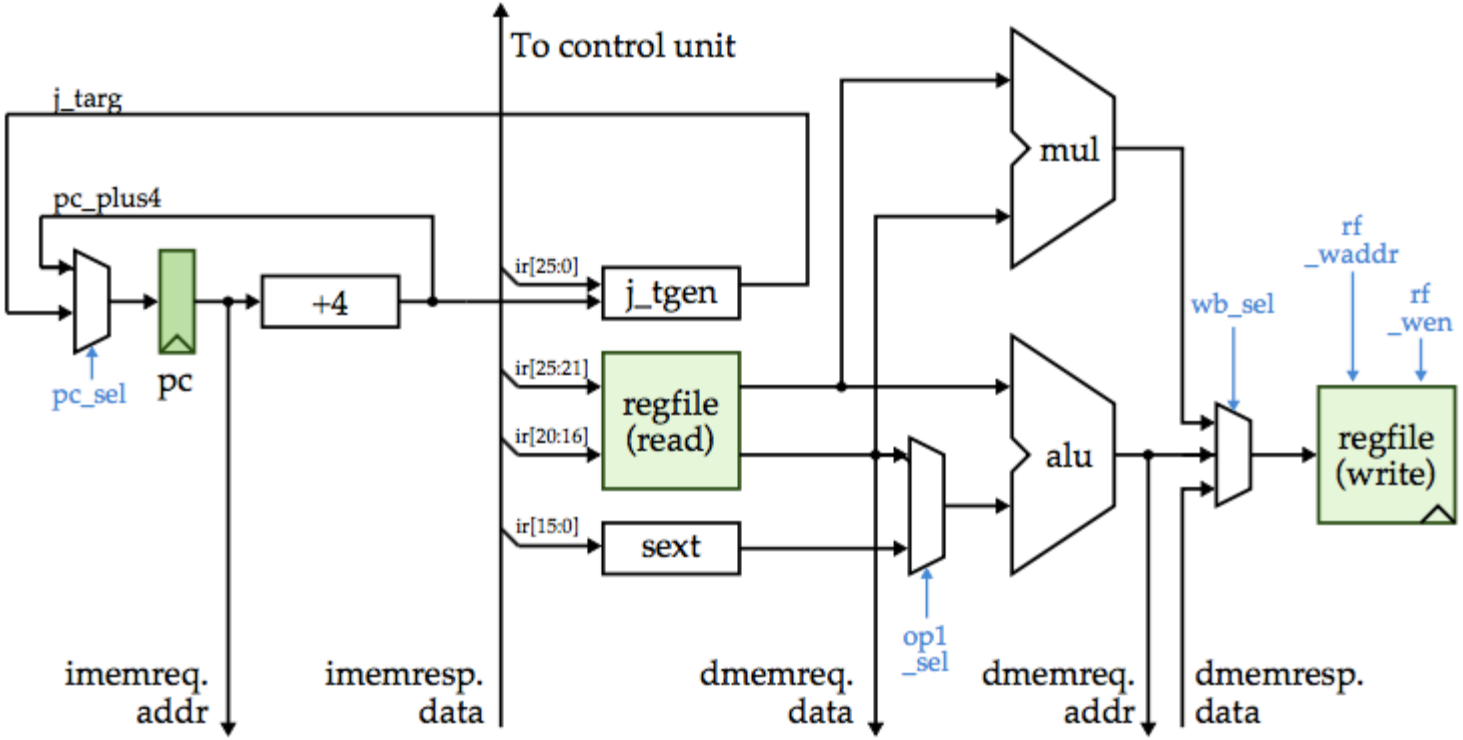
Adding MUL Instruction



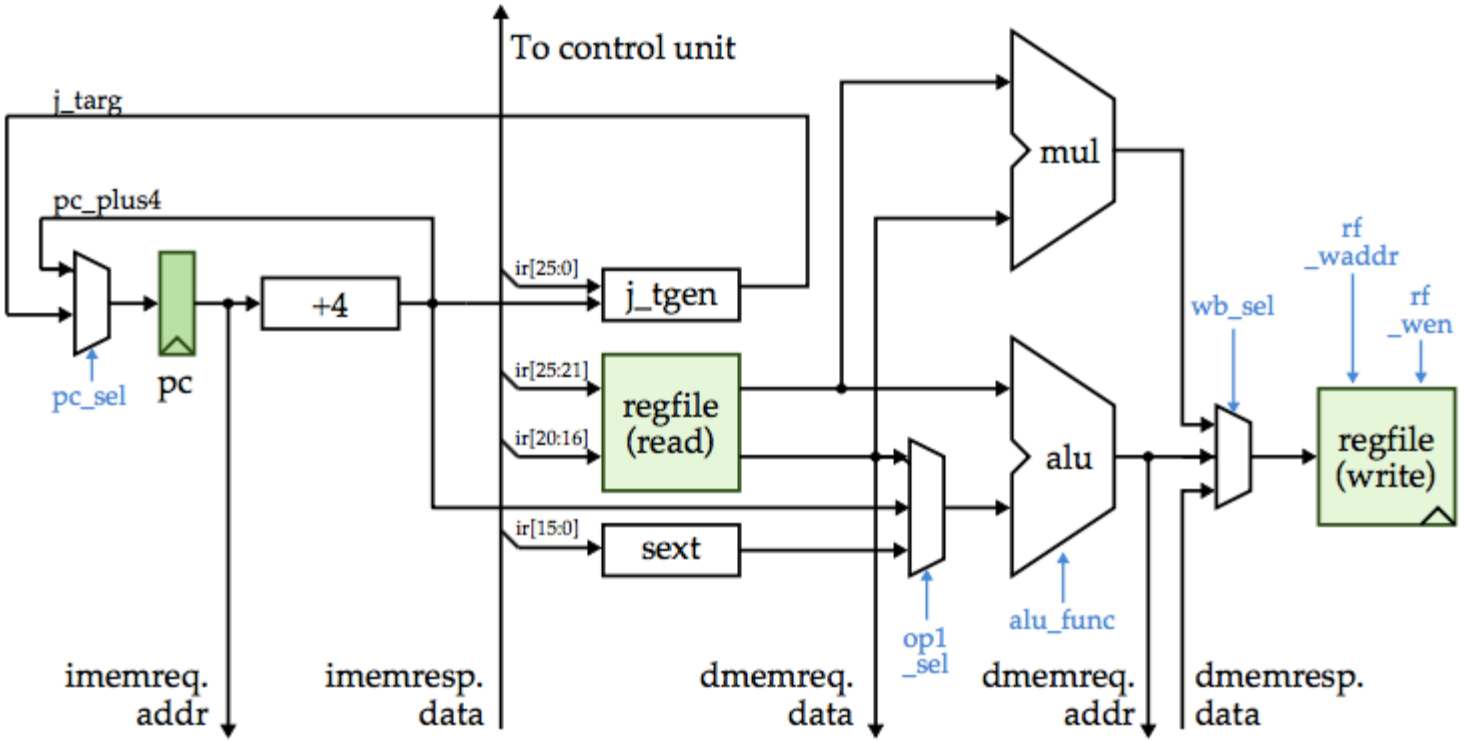
Adding LW and SW Instructions



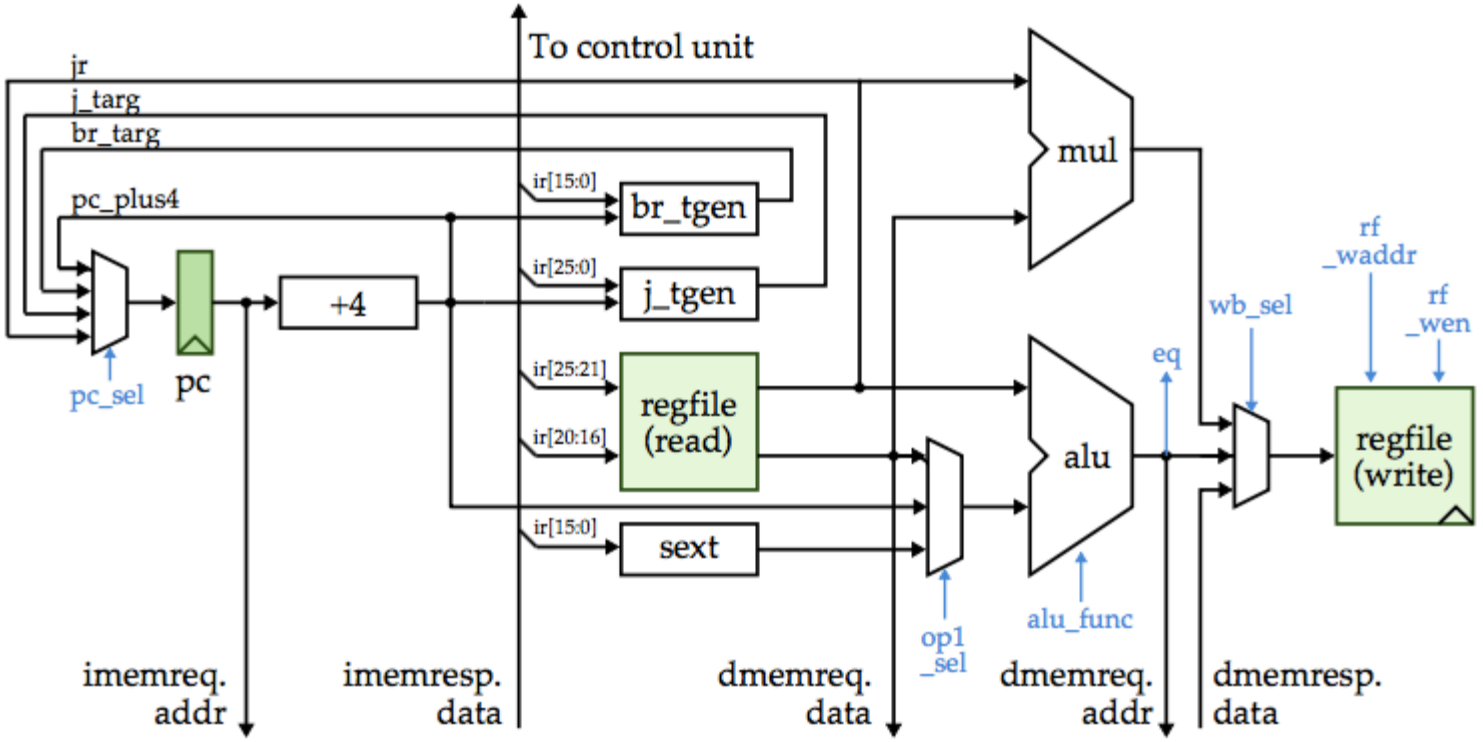
Adding J Instruction



Adding JAL and JR Instructions



Adding BNE Instruction

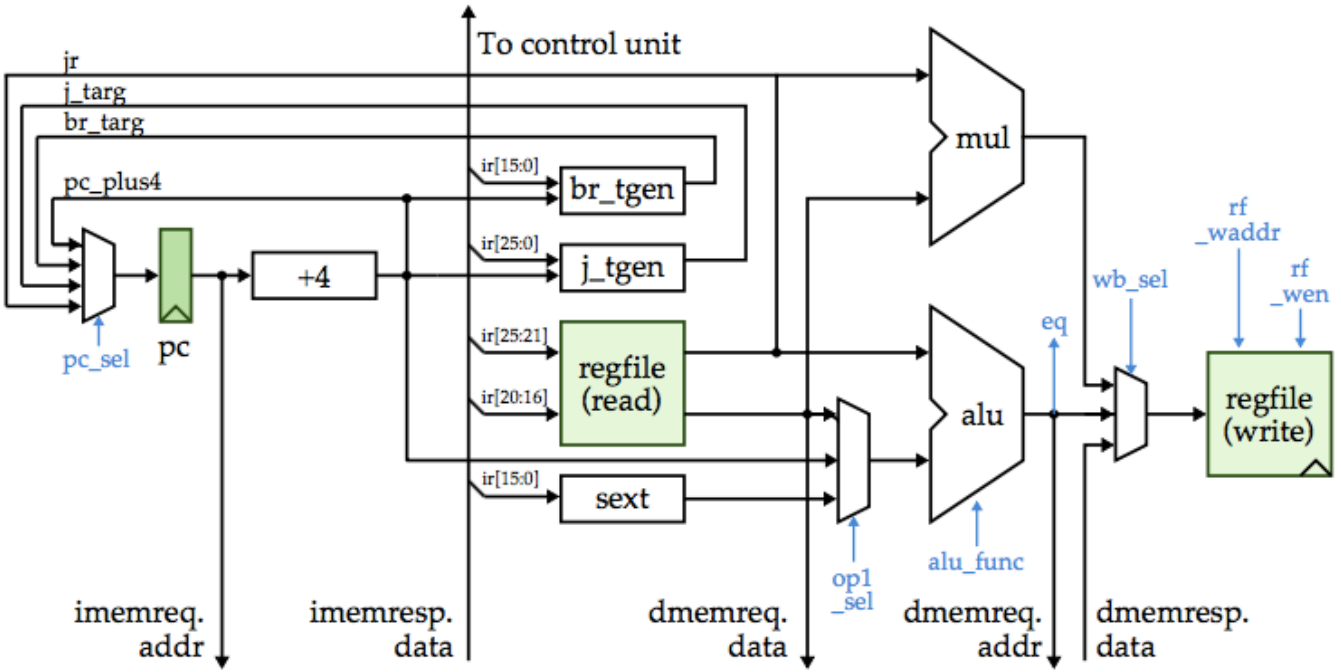


Quiz: Adding a New Auto-Incrementing Load Instruction

Draw on the datapath diagram what paths we need to use as well as any new paths we will need to add in order to implement the following auto-incrementing load instruction.

`lw.ai rt, offset(rs)`

$$R[\texttt{rt}] \leftarrow M[\ R[\texttt{rs}] + \text{sext}(\texttt{offset})\ ];\ R[\texttt{rs}] \leftarrow R[\texttt{rs}] + 4$$

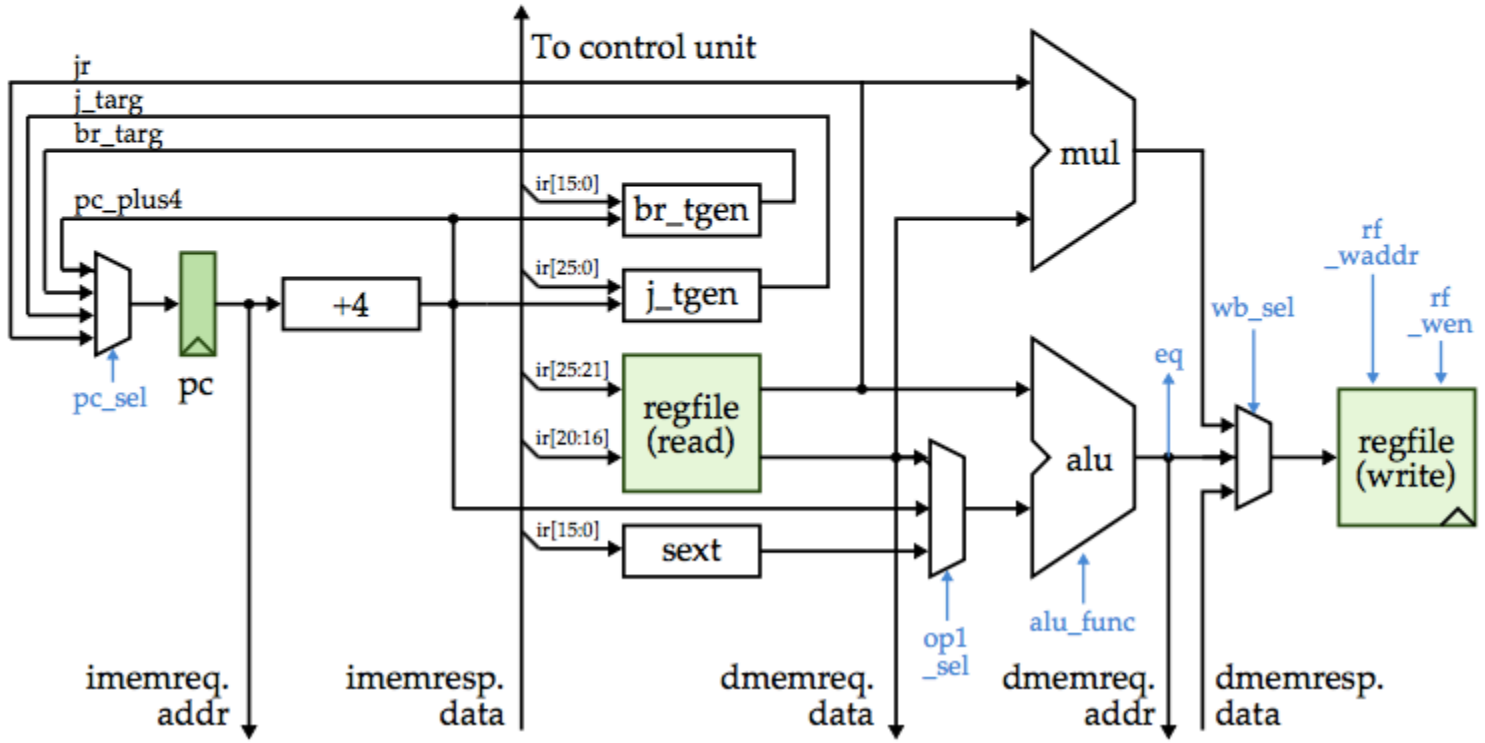


Single-Cycle Processor Control Unit

inst	pc sel	op1 sel	alu func	wb sel	rf waddr	rf wen	imem req val	dmem req val
addu	pc+4	rf	+	alu	rd	1	1	0
addiu								
mul	pc+4	rf	×	mul	rd	1	1	0
lw	pc+4	sext	+	mem	rt	1	1	1
sw								
j	j_targ	–	–	–	–	0	1	0
jal								
jr	jr	–	–	–	–	0	1	0
bne								
lw.ai								

Estimating Cycle Time—Longest Critical Path

There are many paths through the design that start at a state element and end at a state element. The "critical path" is the longest path across all of these paths. We can usually use a simple first-order static timing estimate to estimate the cycle time (i.e., the clock period and thus also the clock frequency).

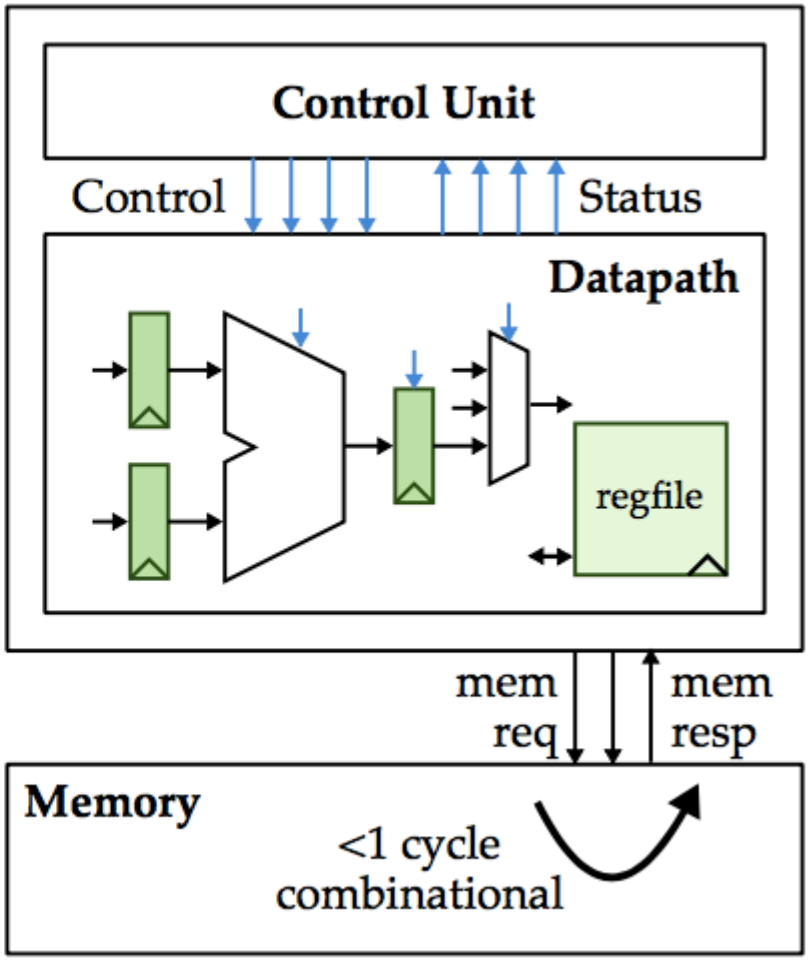


Microarchitecture	CPI	Cycle Time
Single-Cycle Processor	1	long
FSM Processor	>1	short
Pipelined Processor	≈1	short

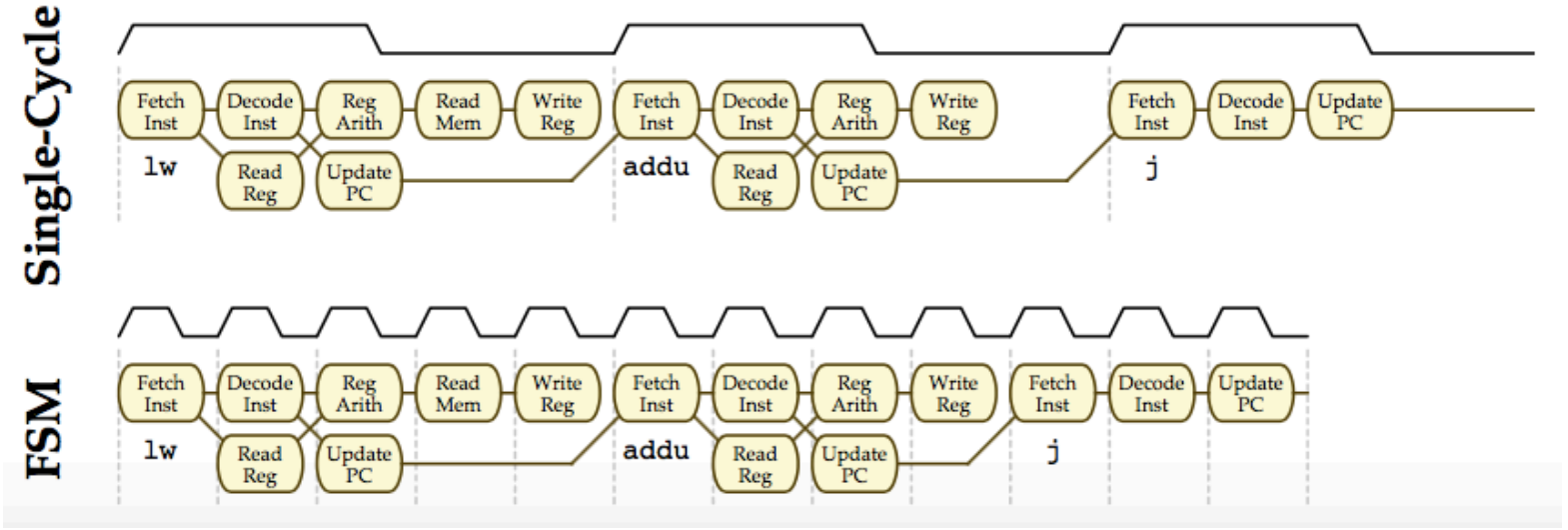
PARCv1 FSM Processor

Technology Constraints

- Assume legacy technology where logic is expensive, so we want to minimize the number of registers and combinational logic
- Assume an (unrealistic) combinational memory
- Assume multi-ported register files and memories are too expensive, these structures can only have a single read/write port

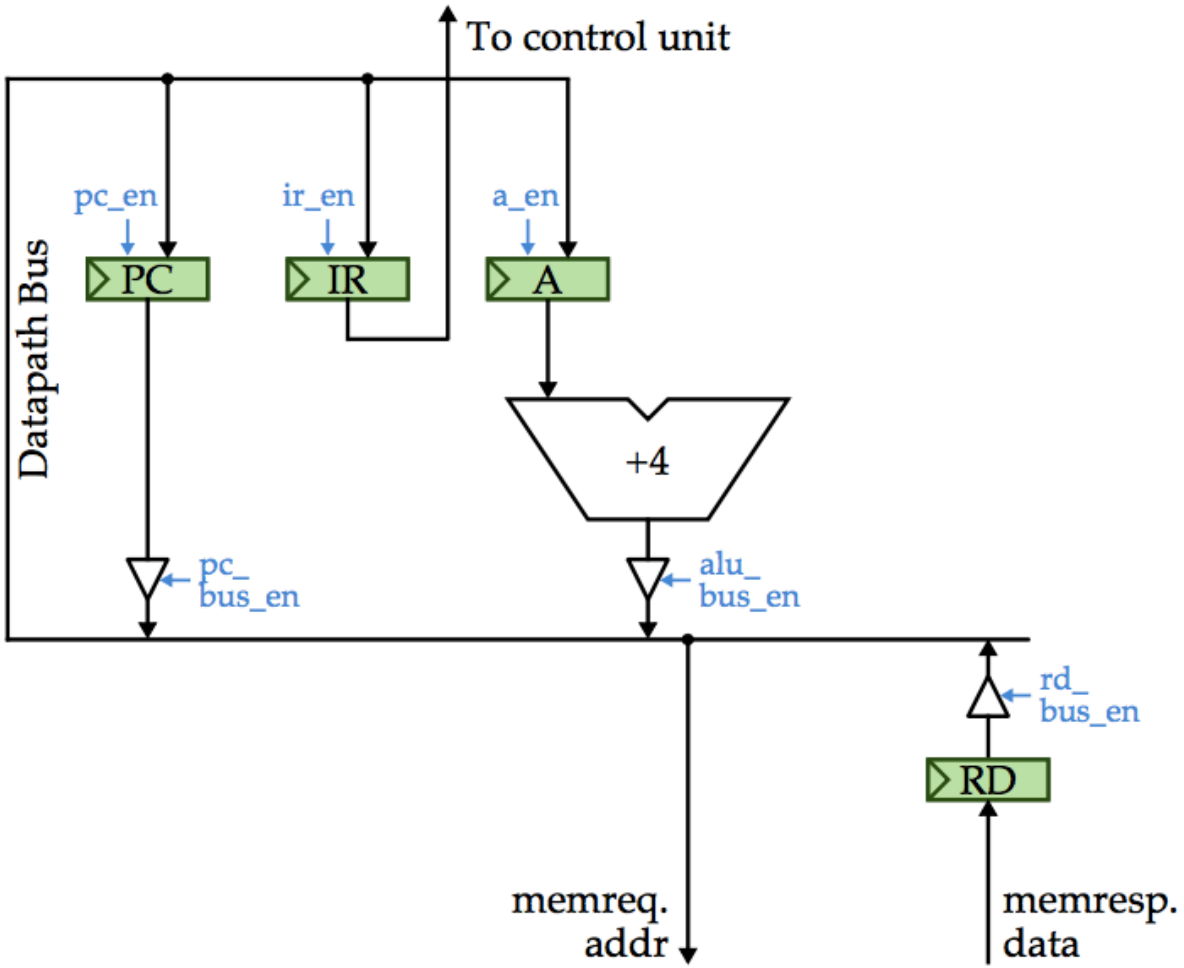


	addu	addiu	mul	lw	sw	j	jal	jr	bne
Fetch Instruction	✓	✓	✓	✓	✓	✓	✓	✓	✓
Decode Instruction	✓	✓	✓	✓	✓	✓	✓	✓	✓
Read Registers	✓	✓	✓	✓	✓			✓	✓
Register Arithmetic	✓	✓	✓	✓	✓				✓
Read Memory				✓					
Write Memory					✓				
Write Registers	✓	✓	✓	✓			✓		
Update PC	✓	✓	✓	✓	✓	✓	✓	✓	✓



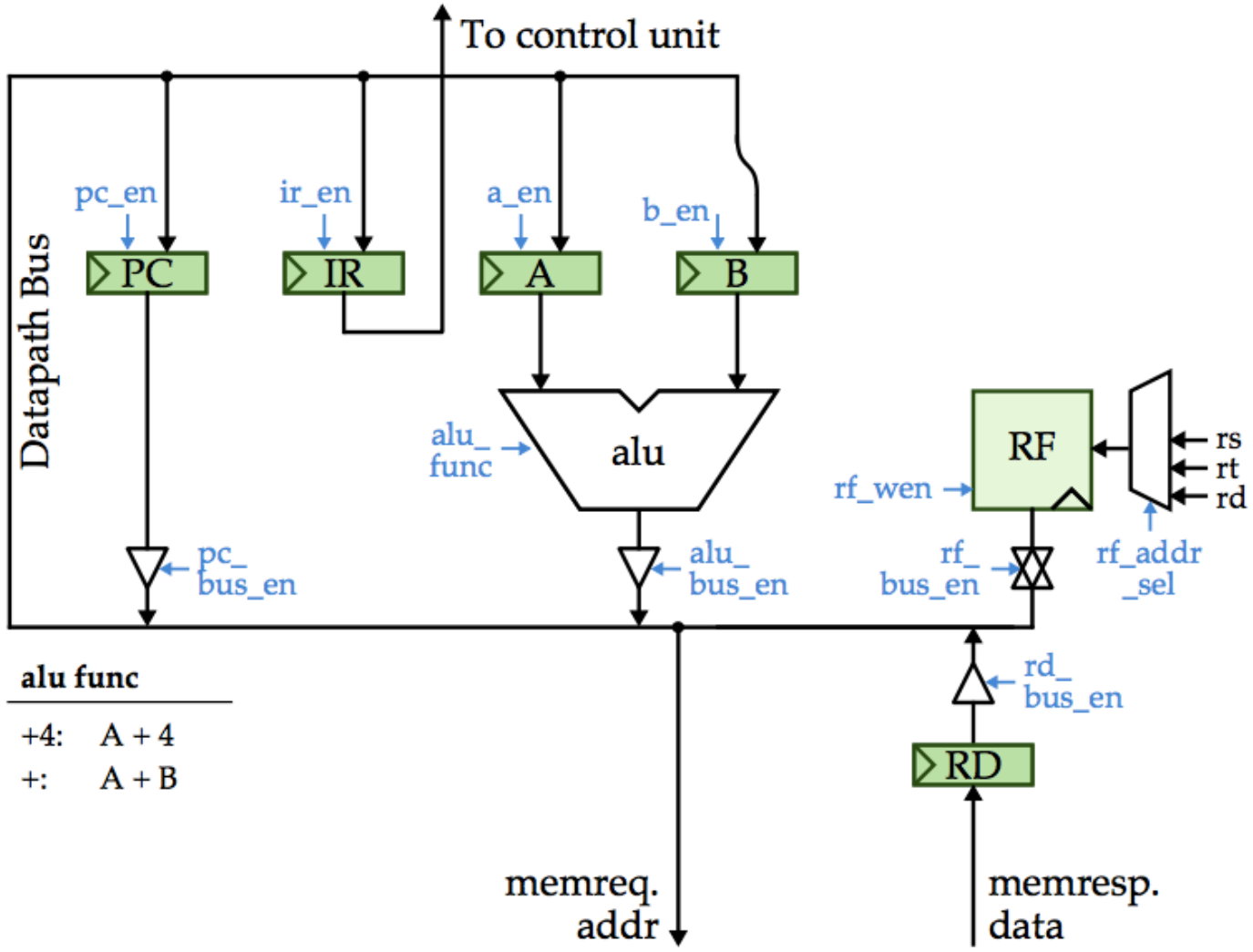
FSM Processor Datapath

- Implement fetch sequence

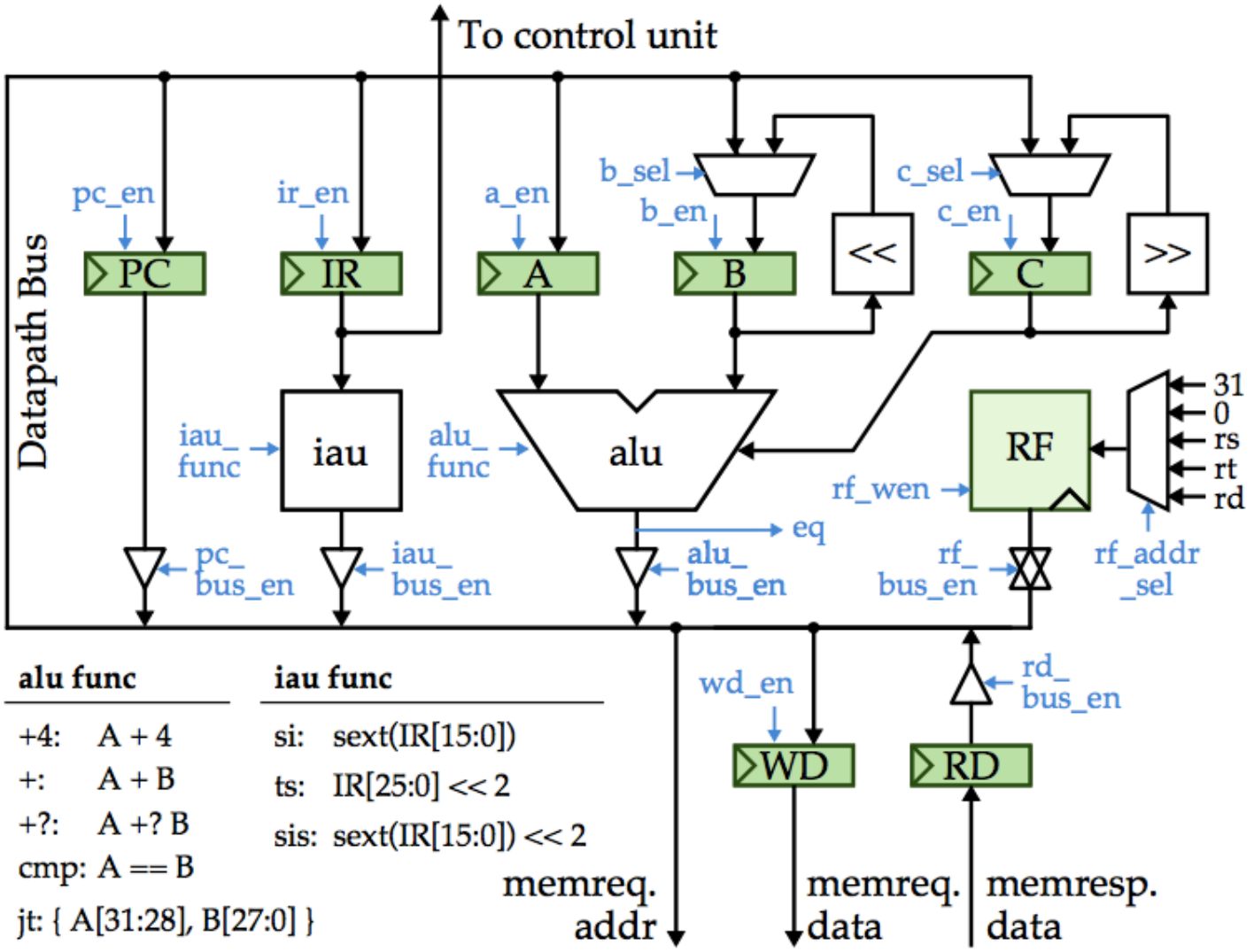


FSM Processor Datapath

- Implement ADDU sequence



Full Datapath for PARCv1 FSM Processor



MUL Instruction

`mul rd, rs, rt`

M0: A ← RF[r0]
M1: B ← RF[rs]
M2: C ← RF[rt]
M3: A ← A +? B;
B ← B << 1; C ← C >> 1
M4: A ← A +? B;
B ← B << 1; C ← C >> 1
...
M35: RF[rd] ← A +? B; goto F0

J Instruction

`j targ`

J0: B ← targ << 2
J1: PC ← A jt B; goto F0

JAL Instruction

`jal targ`

JA0: RF[31] ← PC
JA1: B ← targ << 2
JA2: PC ← A jt B; goto F0

LW Instruction

`lw rt, offset(rs)`

L0: A ← RF[rs]
L1: B ← sext(offset)
L2: memreq.addr ← A + B
L3: RF[rt] ← RD; goto F0

SW Instruction

`sw rt, offset(rs)`

S0: WD ← RF[rt]
S1: A ← RF[rs]
S2: B ← sext(imm)
S3: memreq.addr ← A + B; goto F0

JR Instruction

`jr rs`

JR0: PC ← RF[rs]; goto F0

BNE Instruction

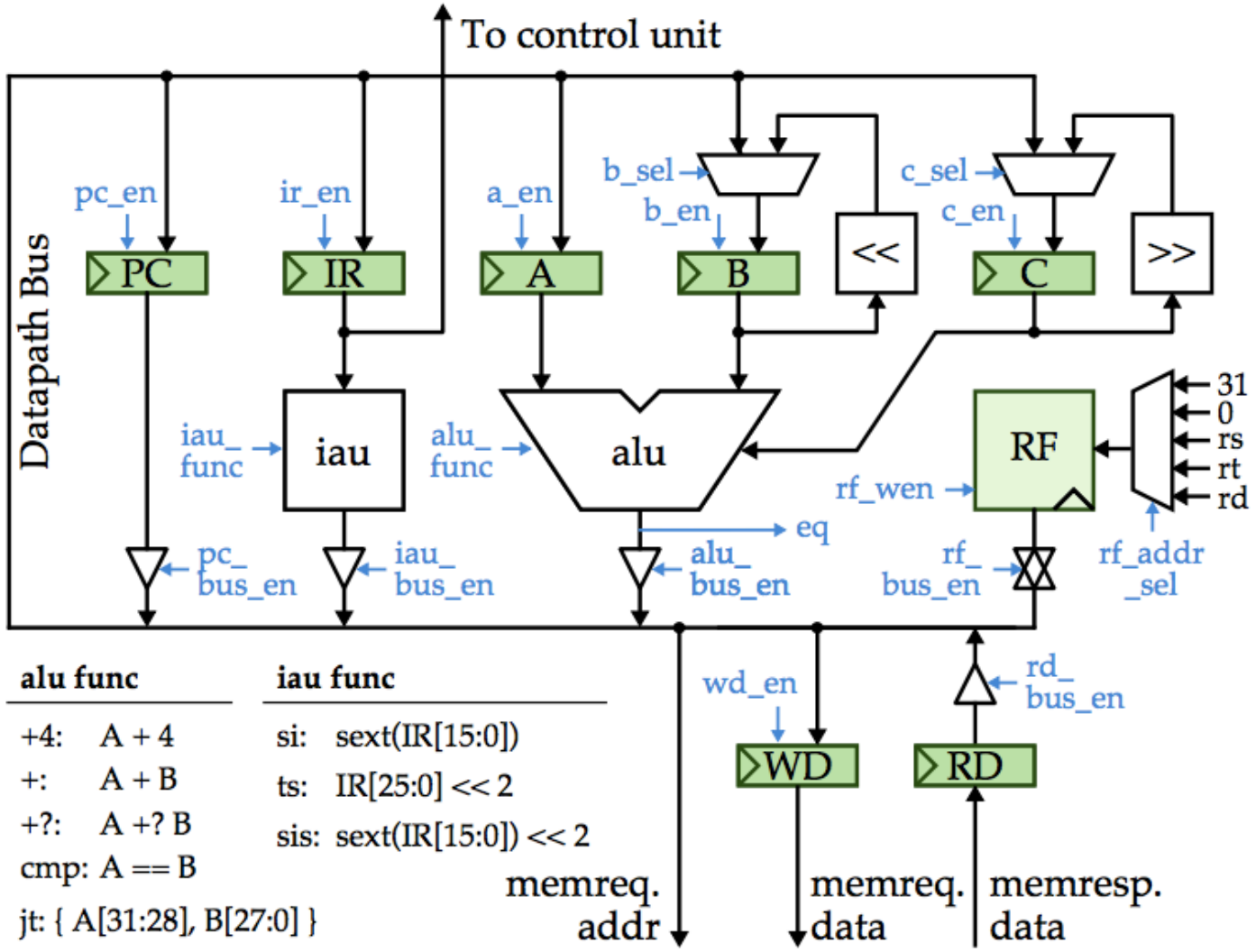
`bne rs, rt, offset`

B0: A ← RF[rs]
B1: B ← RF[rt]
B2: A ← sext(offset) << 2;
    if A == B goto F0
B3: B ← PC
B4: PC ← A + B; goto F0

Adding a Complex Instruction

```
addu.mm rd, rs, rt
```

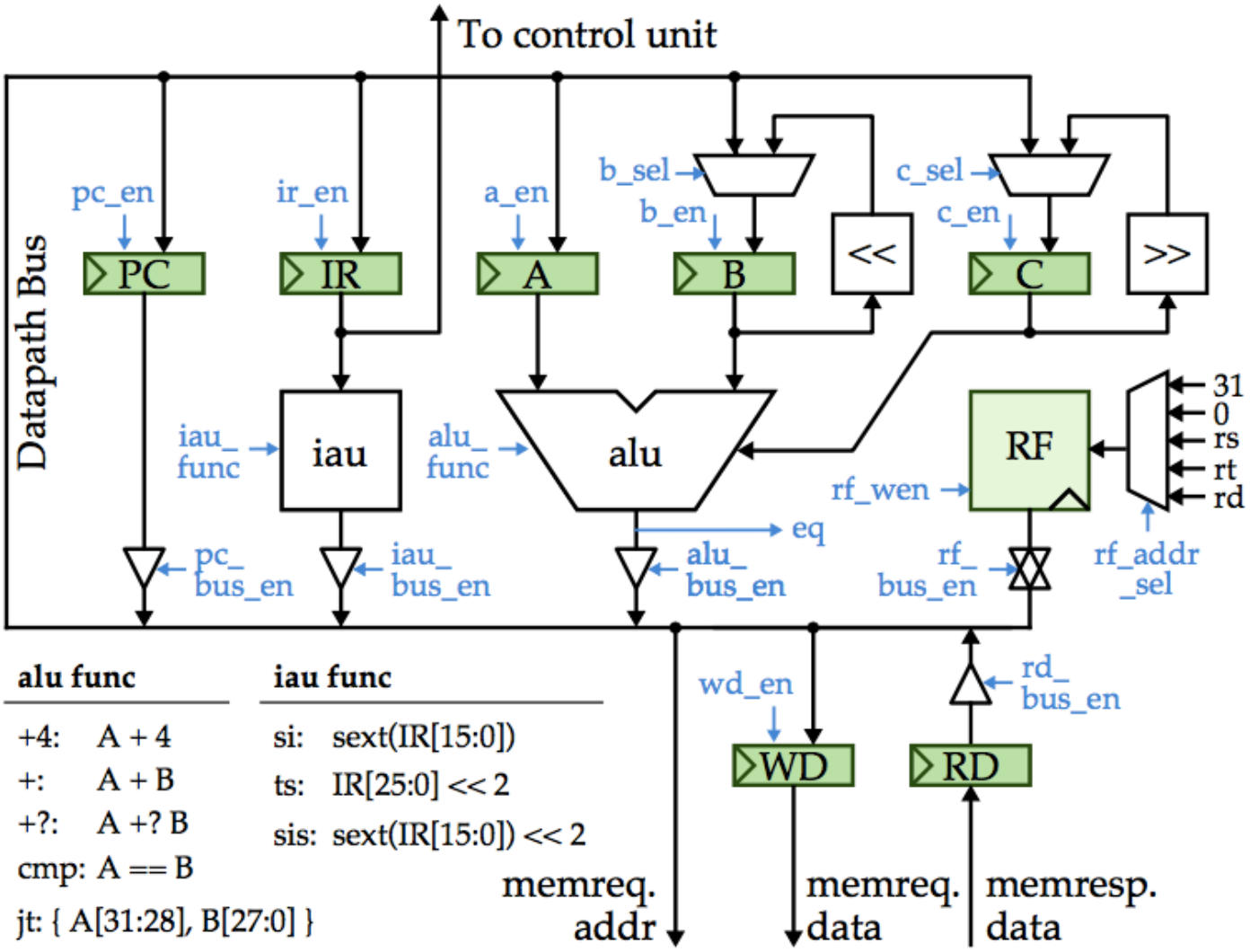
$$M[\,R[rd]\,] \leftarrow M[\,R[rs]\,] + M[\,R[rt]\,]$$



Quiz: Adding a New Auto-Incrementing Load Instruction

`lw.ai rt, offset(rs)`

$$R[\texttt{rt}] \leftarrow M[\, R[\texttt{rs}] + \text{sext}(\texttt{offset})\,];\; R[\texttt{rs}] \leftarrow R[\texttt{rs}] + 4$$



Questions?

Comments?

Discussion?

Acknowledgement

Cornell University, ECE 4750