CISC Design

Performance Enhancement

Virendra Singh
Computer Architecture and Dependable Systems Lab
Department of Electrical Engineering
Indian Institute of Technology Bombay
http://www.ee.iitb.ac.in/~viren/
E-mail: viren@ee.iitb.ac.in

EE-309: Microprocessors

Lecture 27 (28 Sep 2015)

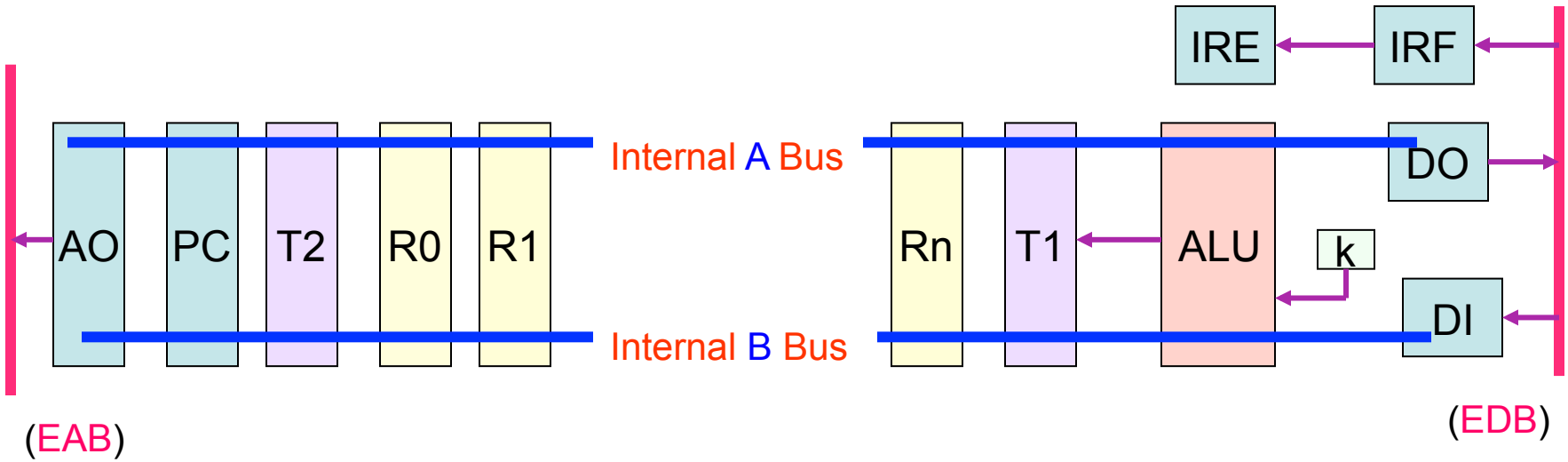
CADSL

MIN Control Word

Control Fields

AO	PC	T2	Regs	T1	ALU	K	DI	DO	IRE	IRF	….

MIN Execution Unit

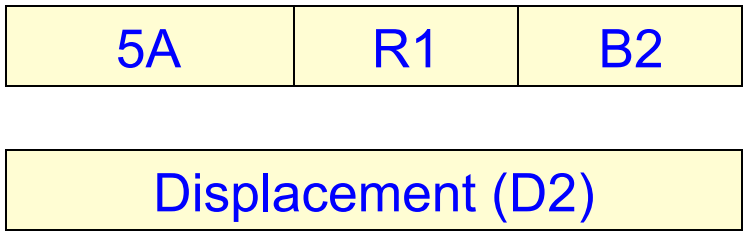


Performance Enhancement

1. More operations
2. Overlap Fetch, Decode, and Execution
3. Balance the operations in a state

ADD Instruction Execution

ADD R1, D2(B2)



Execution Steps

1. Fetch the remaining instruction word
2. Calculate the operand address
3. Fetch the operand
4. Add
5. Store the result
6. Update the program counter
7. Fetch the first half word for the next instruction
8. Find the address of the next instructions control word sequence
9. Branch to the next instruction's control word

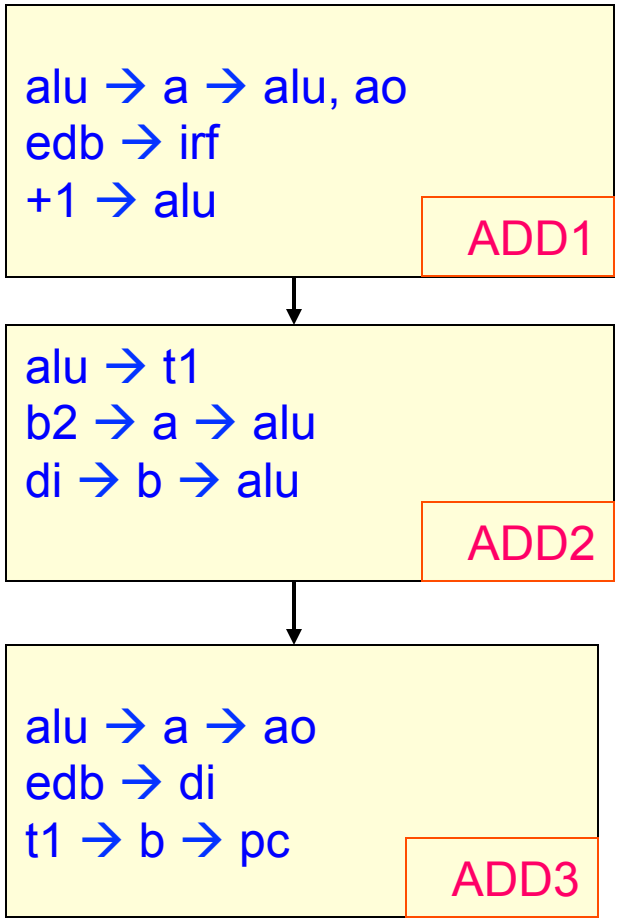
ADD Instruction Execution

State	Execution Unit	Decoder	External Bus
1.			Read instruction half-word
2.	ALU = D2 + (B2)		
3.			Read operand half-word
4.	ALU = (DI) + (R1)		
5.	R1 = (ALU)		
6.	ALU = (PC) + 1		
7.	PC = (ALU)		
8.			Read instruction half-word
9.		IR	

ADD Instruction Execution

State	Execution Unit	Decoder	External Bus
1.	ALU = D2 + (B2)		Read instruction half-word
2.			Read operand half-word
3.	ALU = (DI) + (R1)		
4.	R1 = (ALU)		
5.	ALU = (PC) + 1		
6.	PC = (ALU)	IR	Read instruction half-word

Flowcharts - ADD

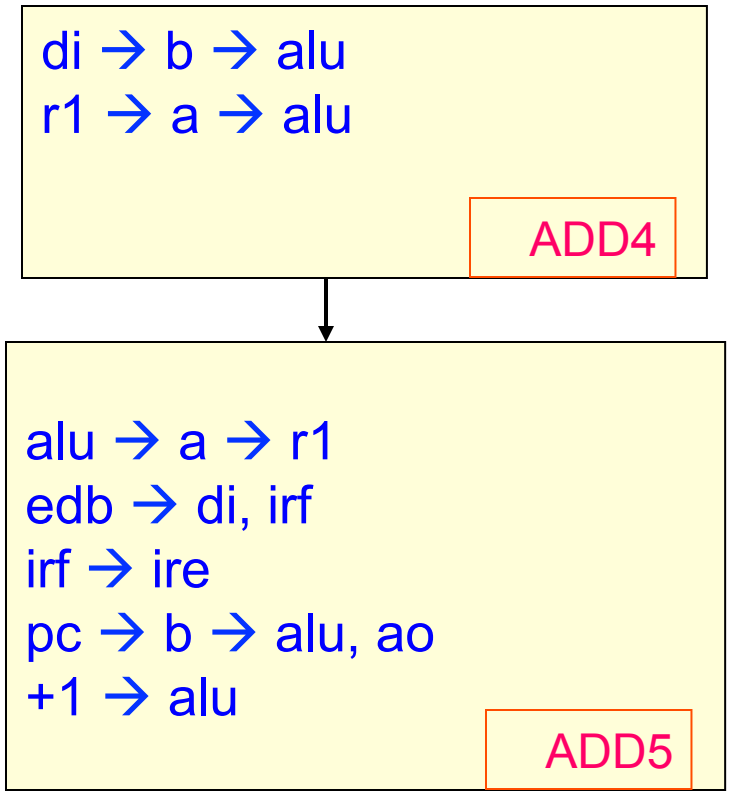


- Increment PC
- Read next instruction half word
- The sum is stored in a register in ALU

- Save the ALU (update PC)
- B2 reg. to internal A bus to ALU
- Displacement D2 from DI to B to ALU
- The sum is stored in a register in ALU

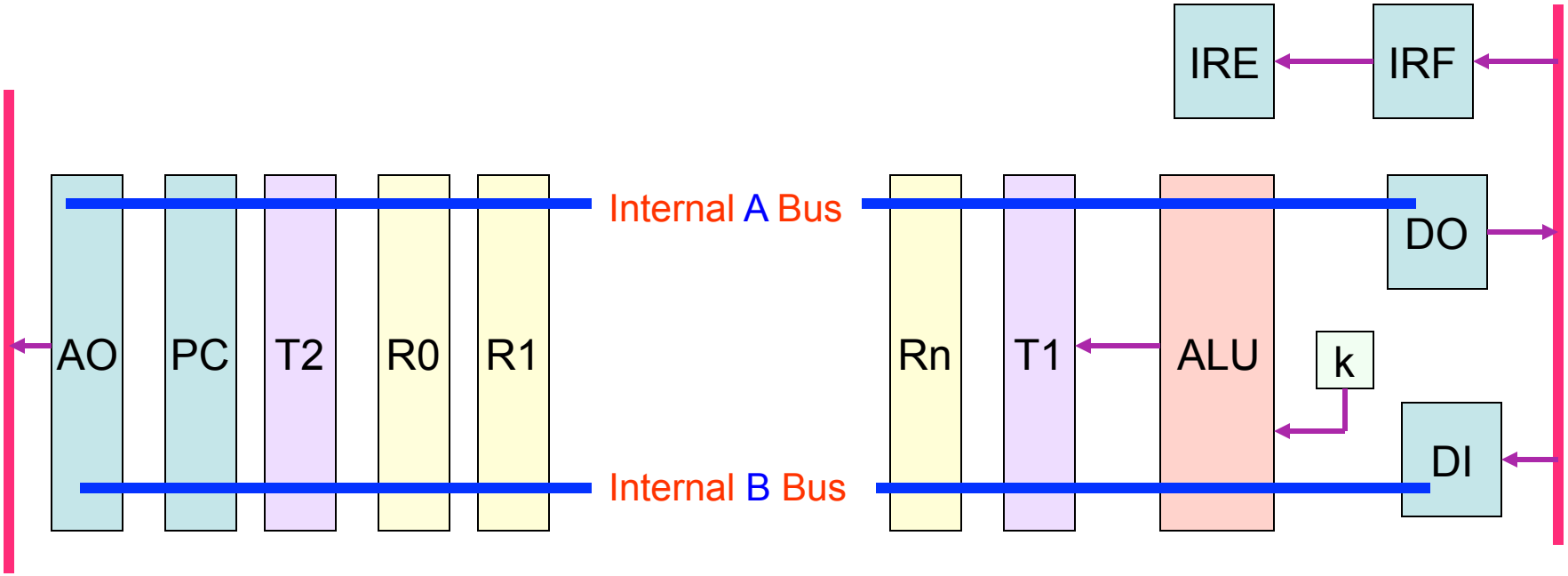
- Send the operand address to the pads
- Read the operand into DI from pads
- Save the updated PC

Flowcharts - ADD

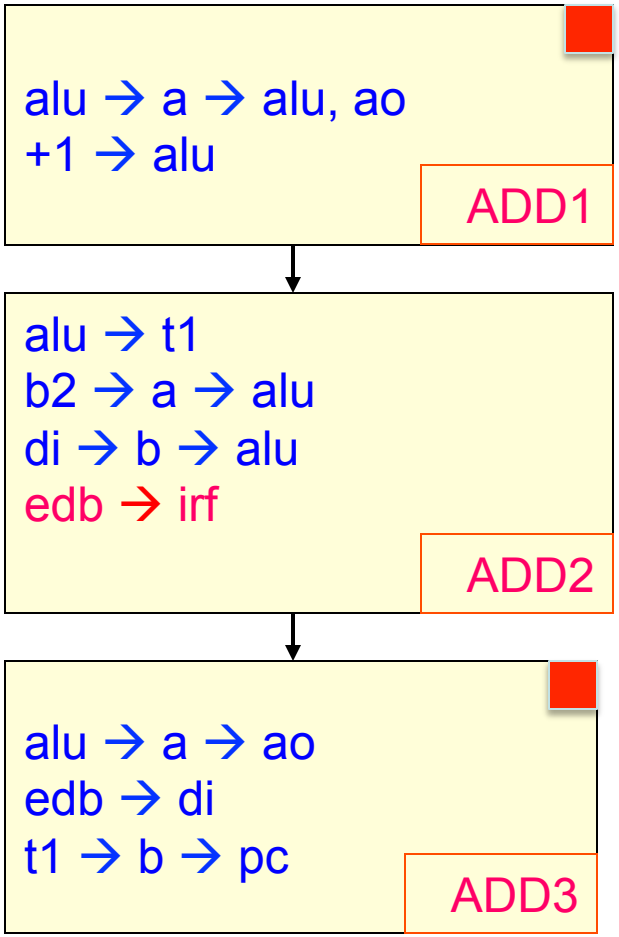


- Add the operand
- Store the result
- Read second instruction halfword
- Decode the next instruction
- Update PC & read instruction halfword

MIN Datapath



Memory Cycle Minimization

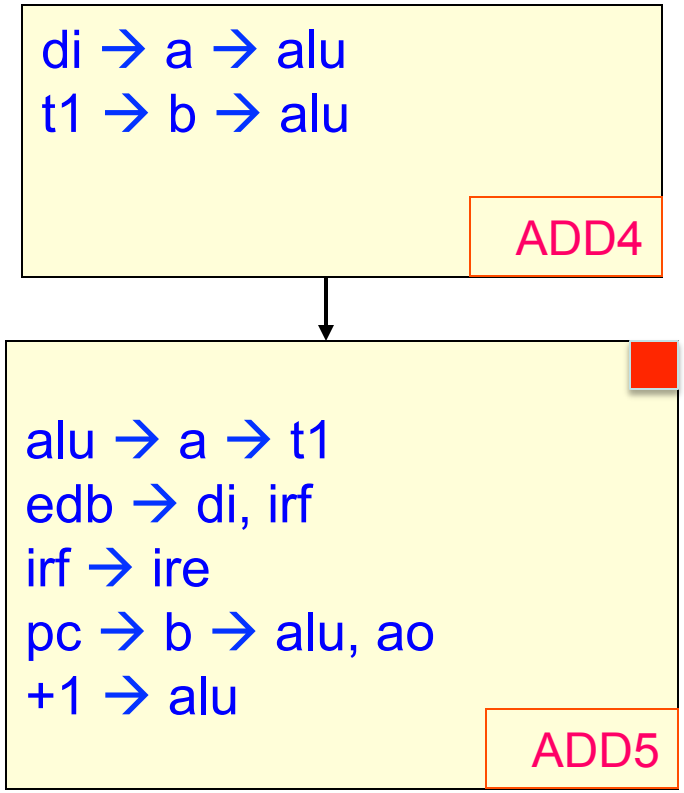


- Increment PC/ initiate instr. read
- The sum is stored in a register in ALU

- Save the ALU (update PC)
- B2 reg. to internal A bus to ALU
- Displacement D2 from DI to B to ALU
- Read next instr. from external bus

- Send the operand address to the pads
- Read the operand into DI from pads
- Save the updated PC

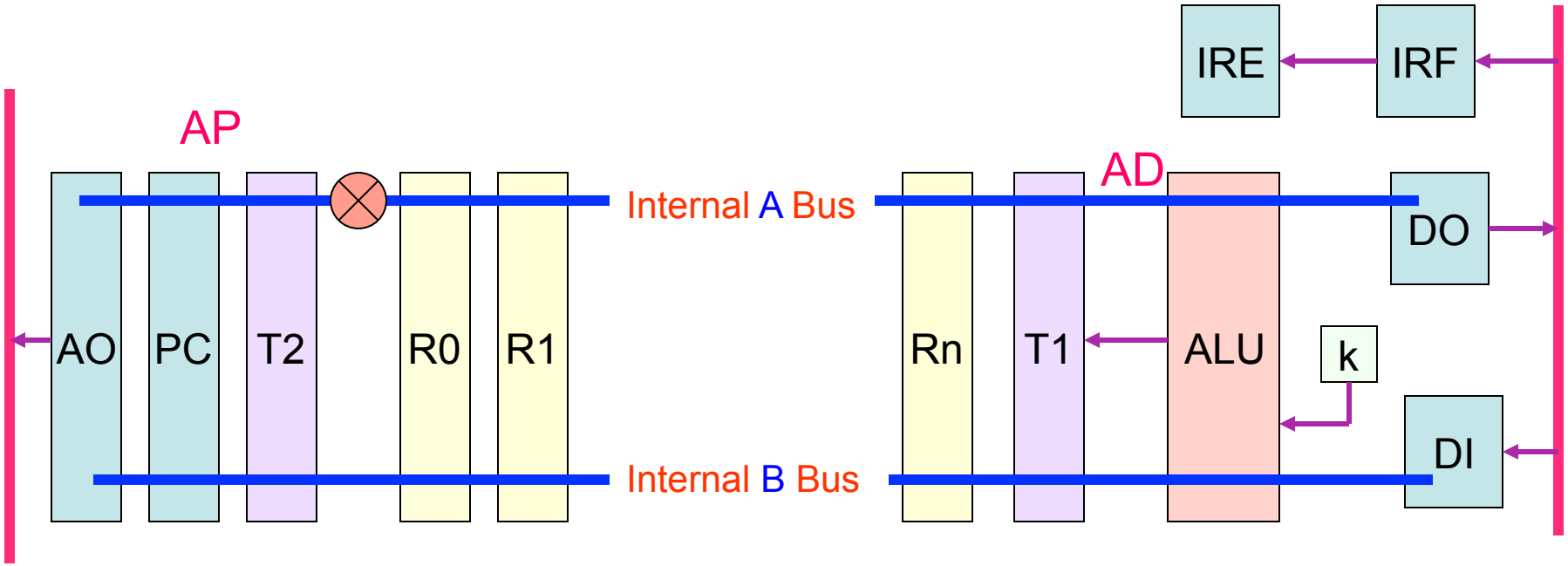
Memory Cycle Minimization



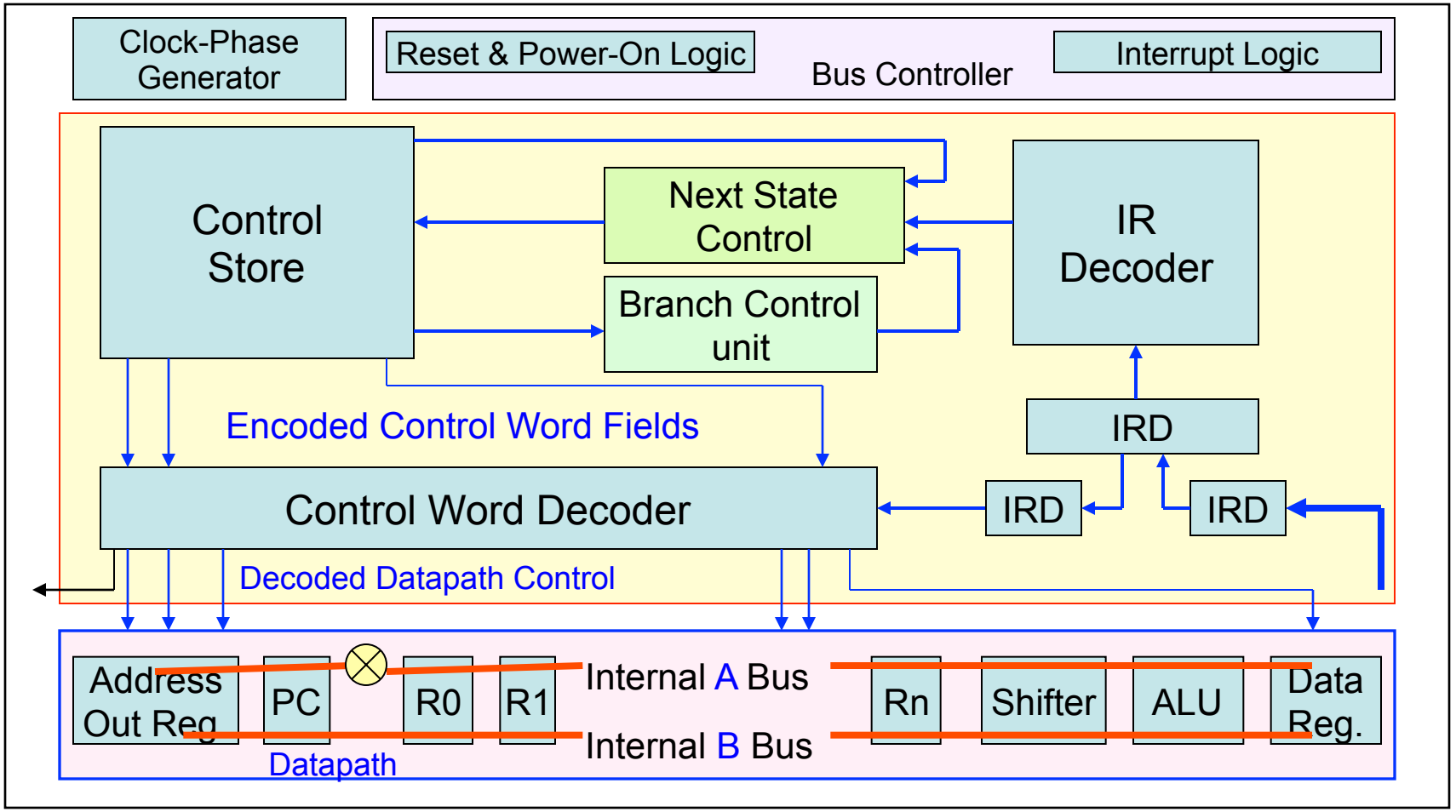
- Add the operand
- The sum is stored in the ALU O/P reg.

- Store the result
- Read second instruction half-word
- Decode the next instruction
- Update PC & read instr. half-word

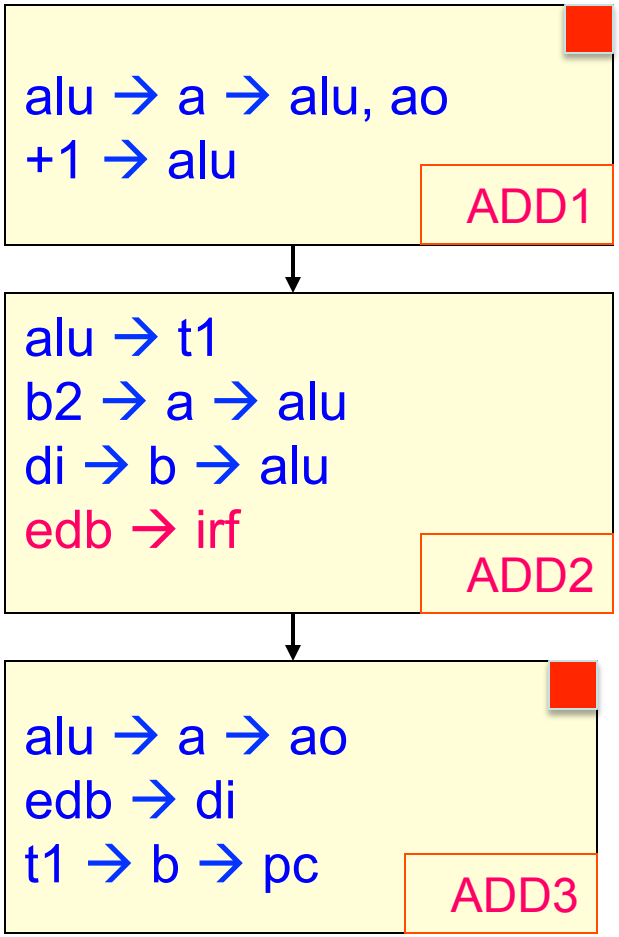
Segmented Bus



Processor - Block Diagram



Memory Cycle Minimization

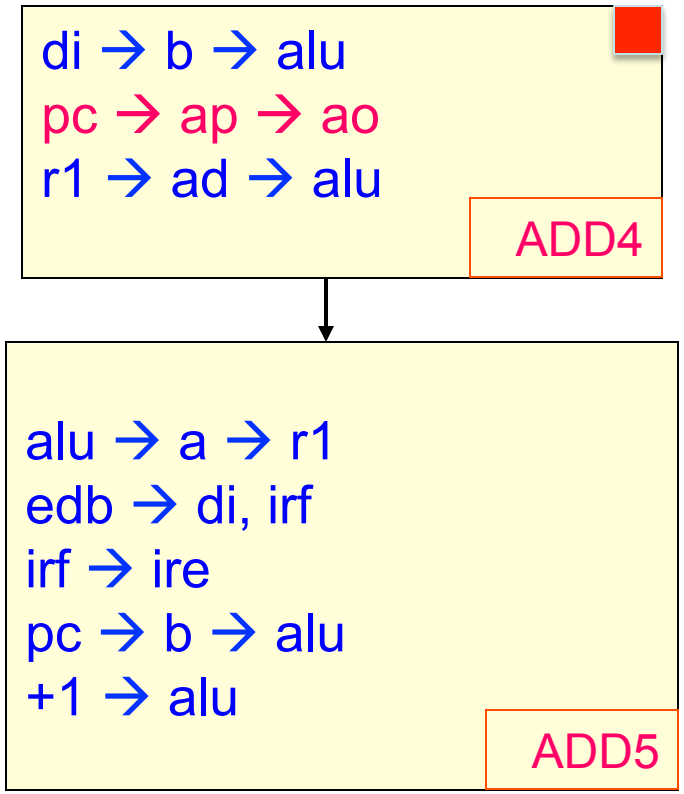


- Increment PC/ initiate instr. read
- The sum is stored in a register in ALU

- Save the ALU (update PC)
- B2 reg. to internal A bus to ALU
- Displacement D2 from DI to B to ALU
- Read next instr. from external bus

- Send the operand address to the pads
- Read the operand into DI from pads
- Save the updated PC

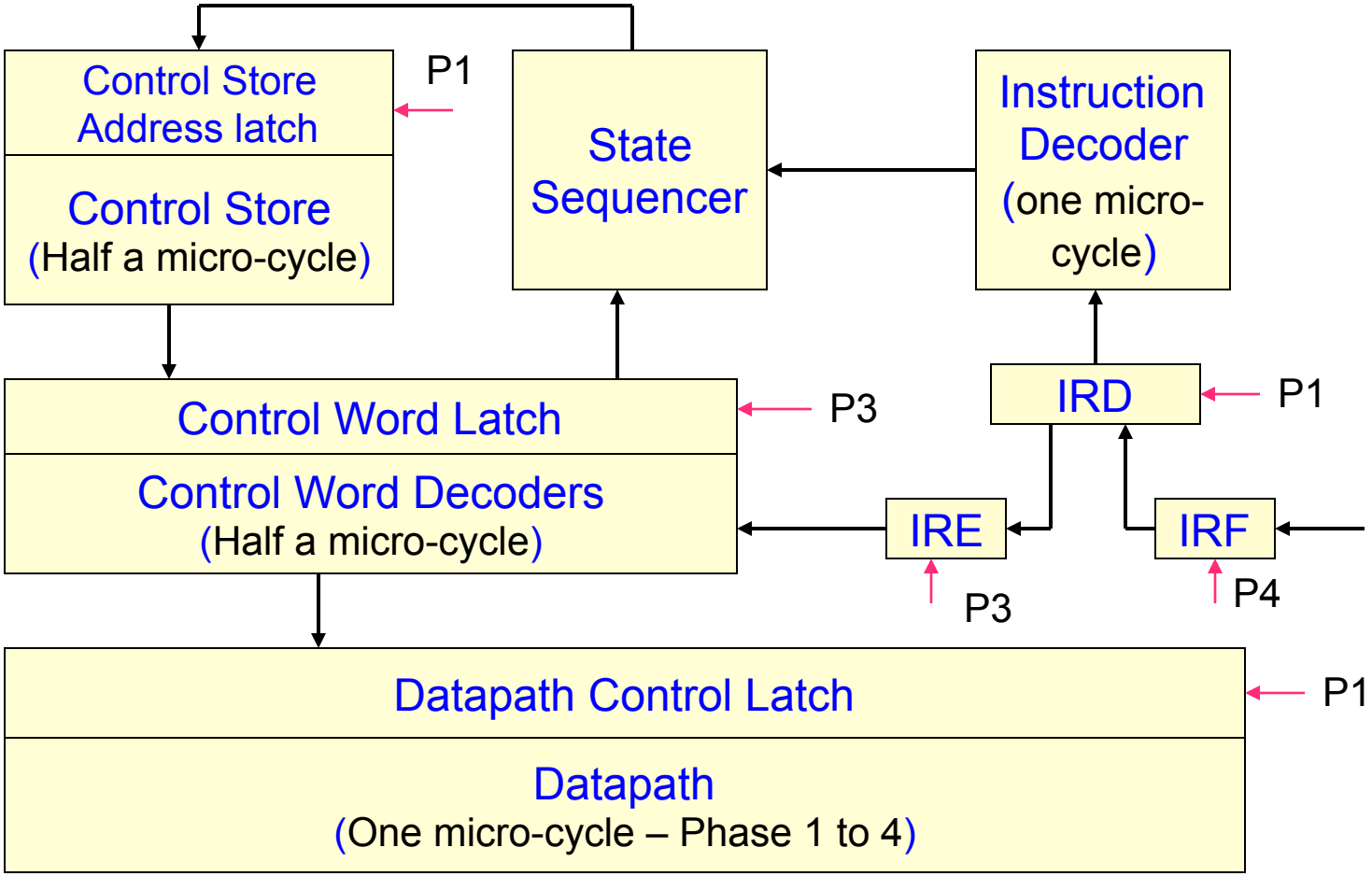
Memory Cycle Minimization



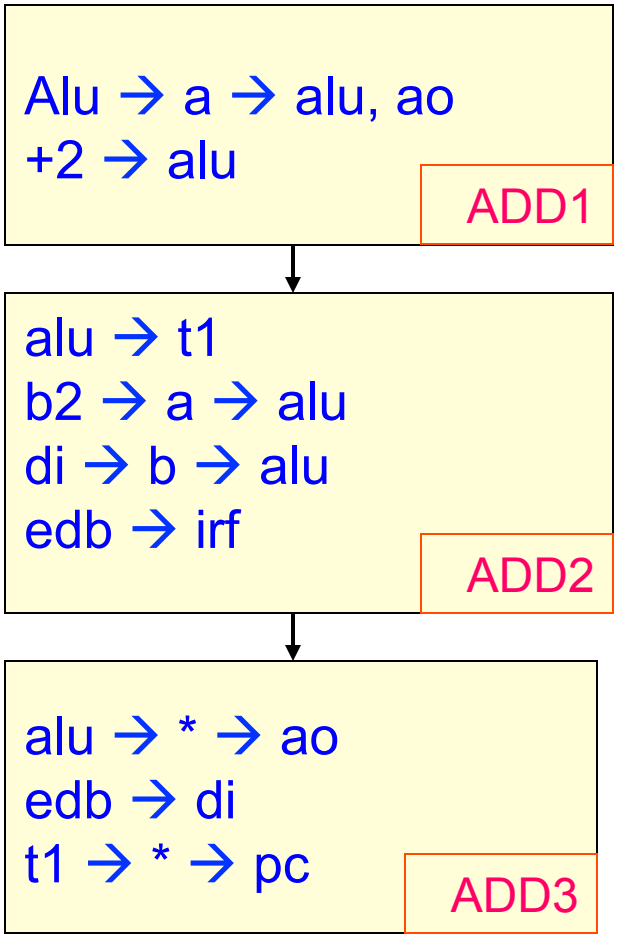
- Add the operand
- Initiate read for next instr. half-word
- The sum is stored in the ALU o/p reg.

- Store the result
- Read second instruction half-word
- Decode the next instruction
- Update PC & read instr. half-word

Internal Timing



Flowcharts - ADD

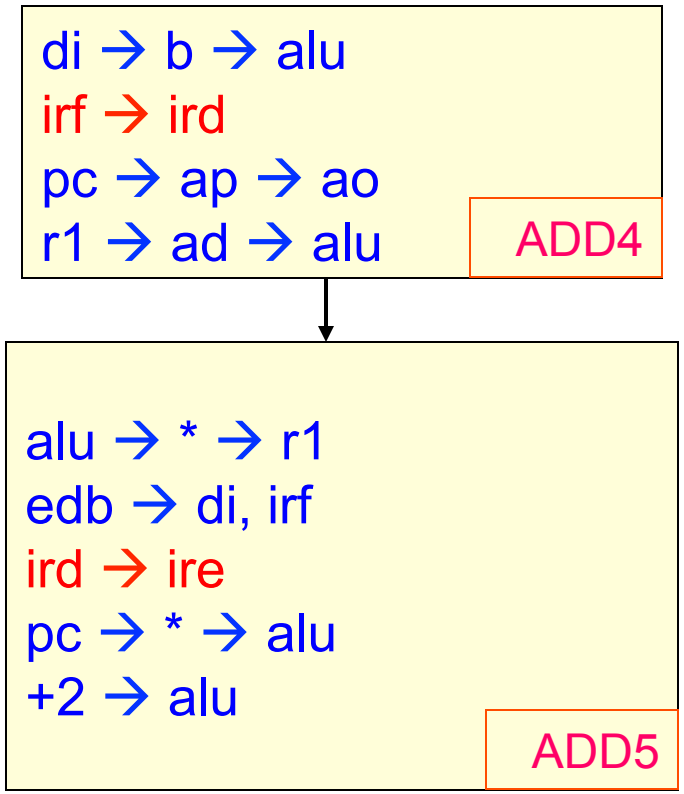


- Increment PC/ initiate instr. read
- The sum is stored in a register in ALU

- Save the ALU (update PC)
- B2 reg. to internal A bus to ALU
- Displacement D2 from DI to B to ALU
- Read next instr. Half word

- Send the operand address to the pads
- Read the operand into DI from pads
- Save the updated PC

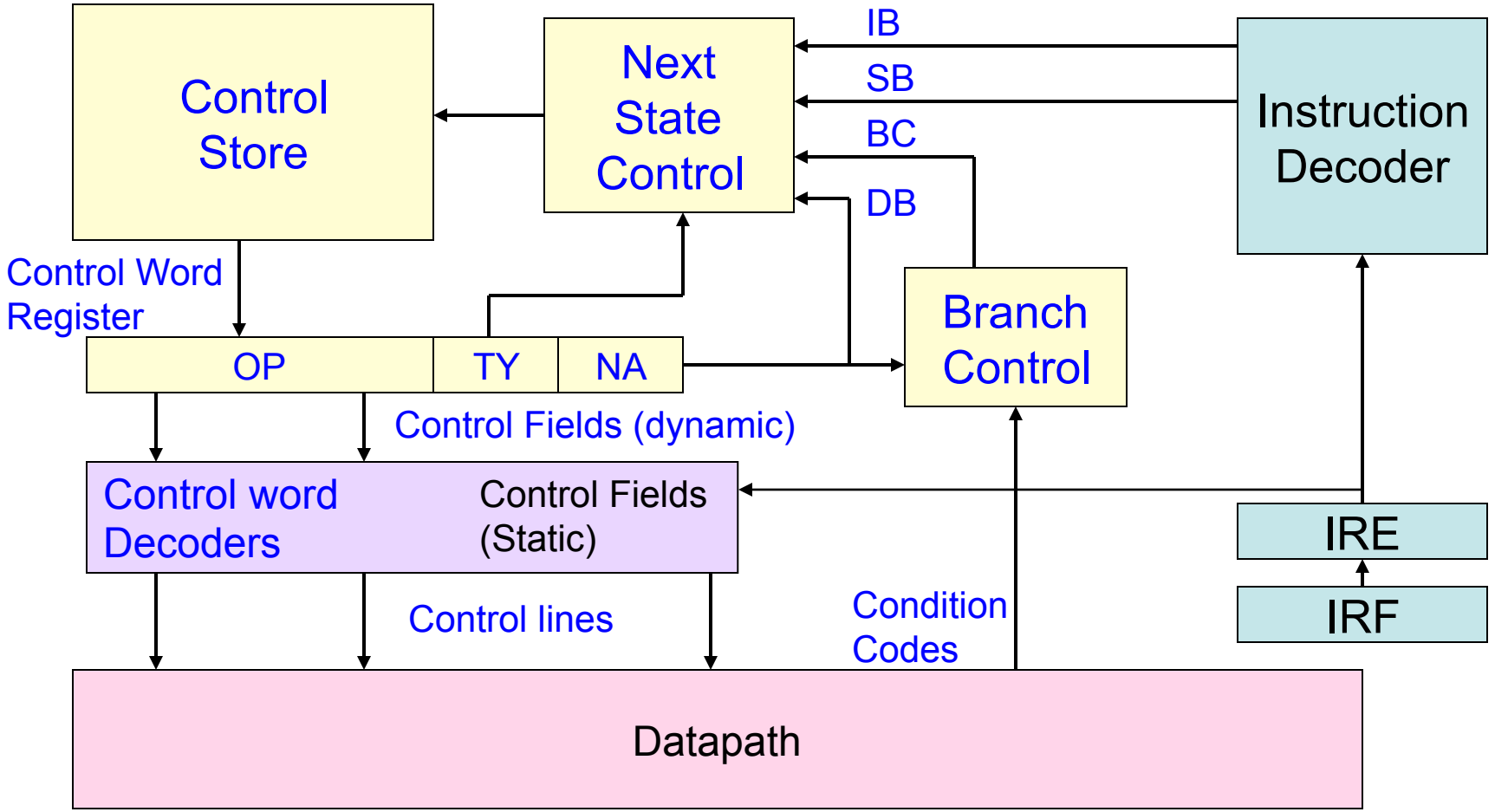
Flowcharts - ADD



- Add the operand
- Decode next instruction
- Initiate read for next instr. half-word
- The sum is stored in the ALU o/p reg.

- Store the result
- Read second instruction half-word
- Decode the next instruction
- Update PC & read instr. half-word

Processor Block Diagram



Thank You

MIN Instruction Set

- ✓ ADD
- ✓ AND
- ✓ BZ – Branch if zero bit is set. (Register Indirect only)
- ✓ LOAD – Second operand is source and Rx is destination
- ✓ POP – Postincrement with register indirect only
- ✓ PUSH – Predecrement with register indirect only
- ✓ STORE
- ✓ SUB
- ✓ TEST