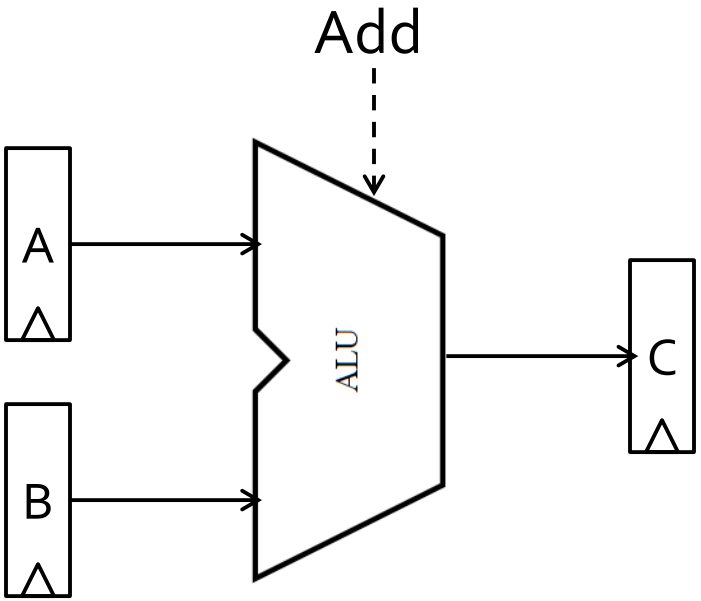
CSCE 230, Fall 2013

Chapter 5: Basic Processing Unit
Part 1: Fundamental Concepts and Datapath Design

Mehmet Can Vuran, Instructor
University of Nebraska-Lincoln

Acknowledgement: Overheads adapted from those provided by the authors of the textbook

Hardware Implementation of A = B+C

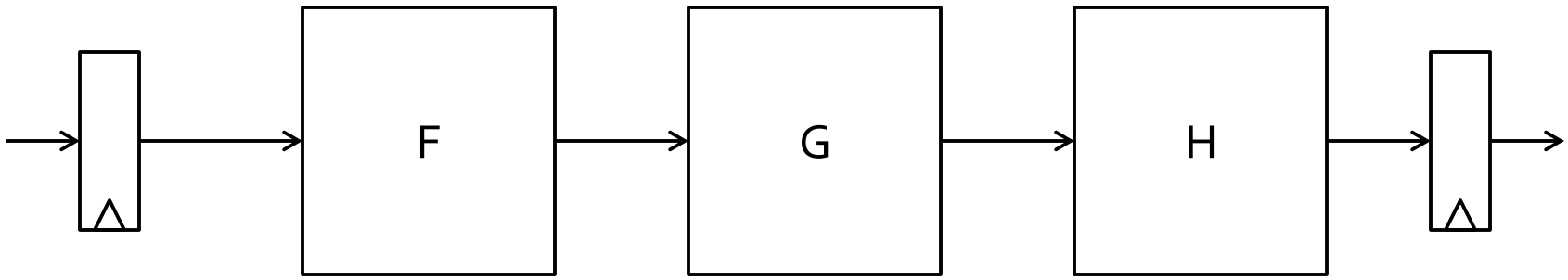


- Common clock
- Operation takes one clock cycle.
- Clock period* ≥
  Propagation_Delay(A or B) +
  Propagation_Delay(ALU) +
  Setup_Time(C)

$$\Leftrightarrow D_{(A \text{ or } B)} + D_{ALU} + D_{setup(C)}$$

*This computation ignores wire delays which can be quite significant, especially, in FPGA implementations

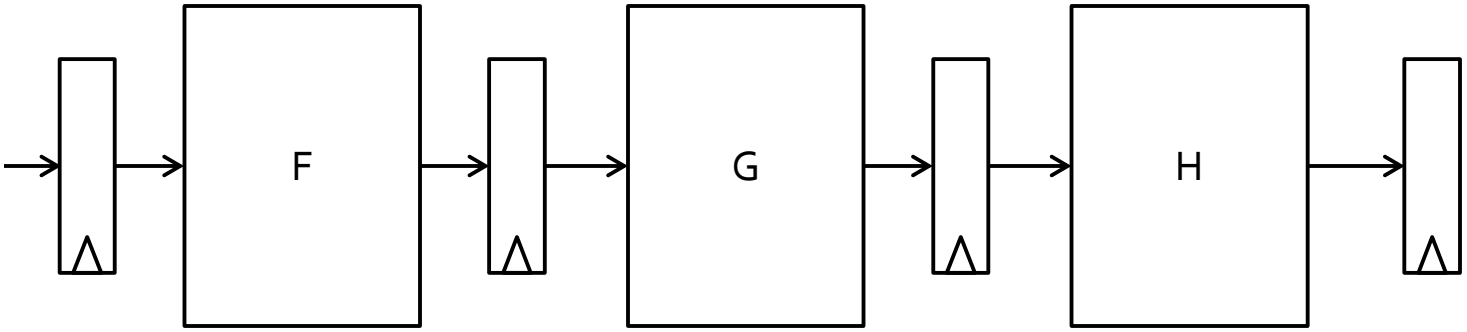
Generalization to Composition of Combinational-Logic Functions



$$\text{Clock Period} \geq \Delta_{(REG)} + \Delta_F + \Delta_G + \Delta_H + \Delta_{setup(REG)}$$

Pipelining computations

Add buffers between stages to achieve independence.



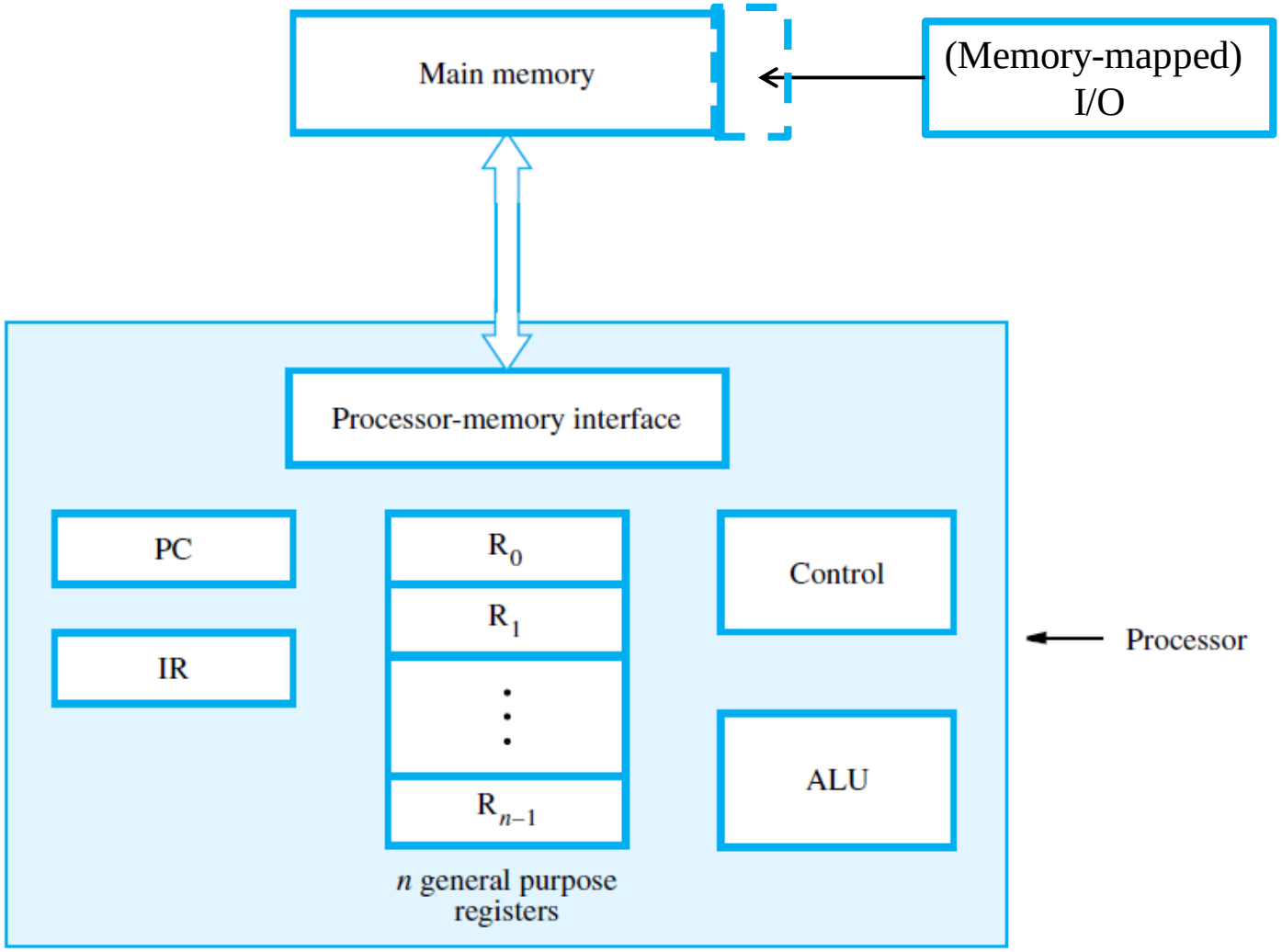
$$\text{Clock Period} \geq D_{(REG)} + Max(D_F, D_G, D_H) + D_{setup(REG)}$$

Again, ignoring wire delays

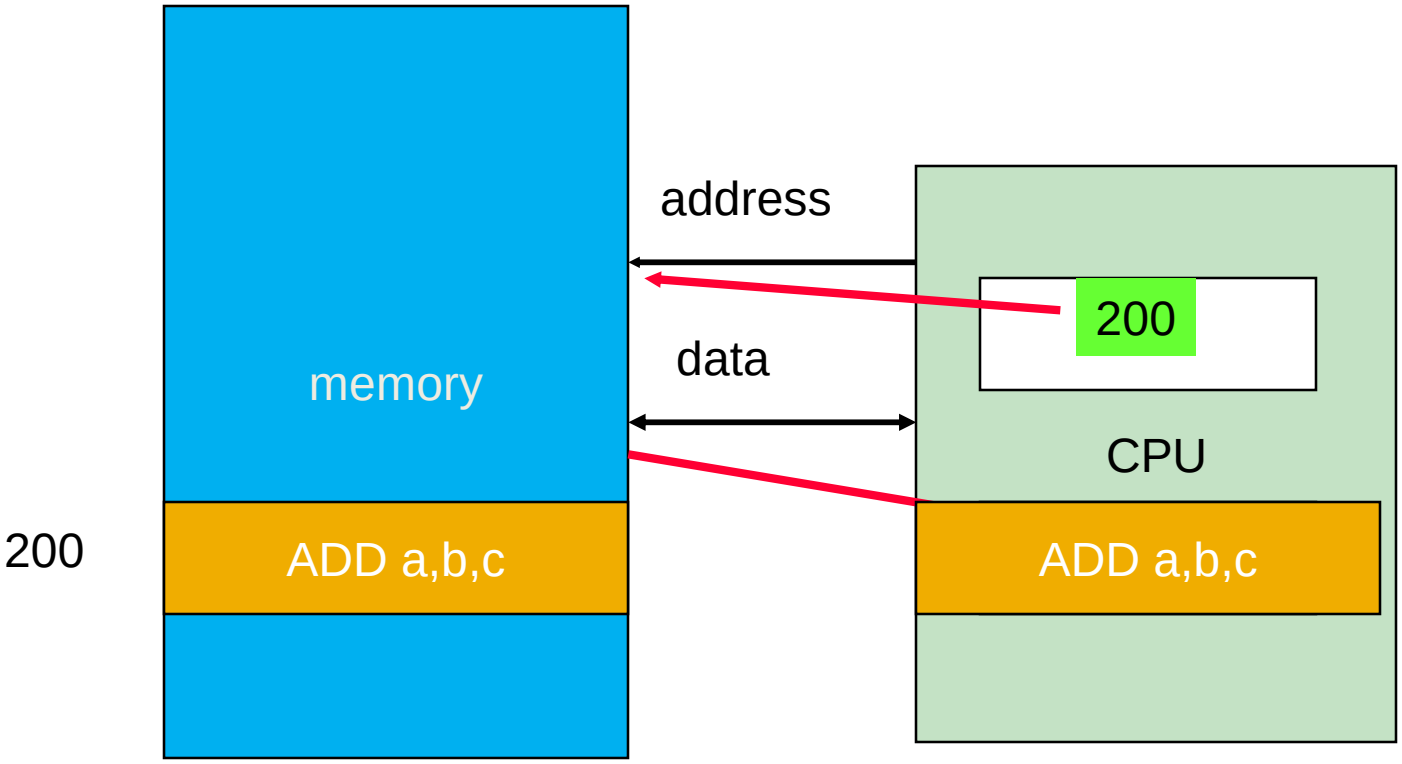
A note on processor of Chapter 5

- The design uses inter-stage buffers
- Motivation:
  - Easier-to-understand control
  - Easier to generalize to pipelined implementation in Chapter 6

Computer Architecture

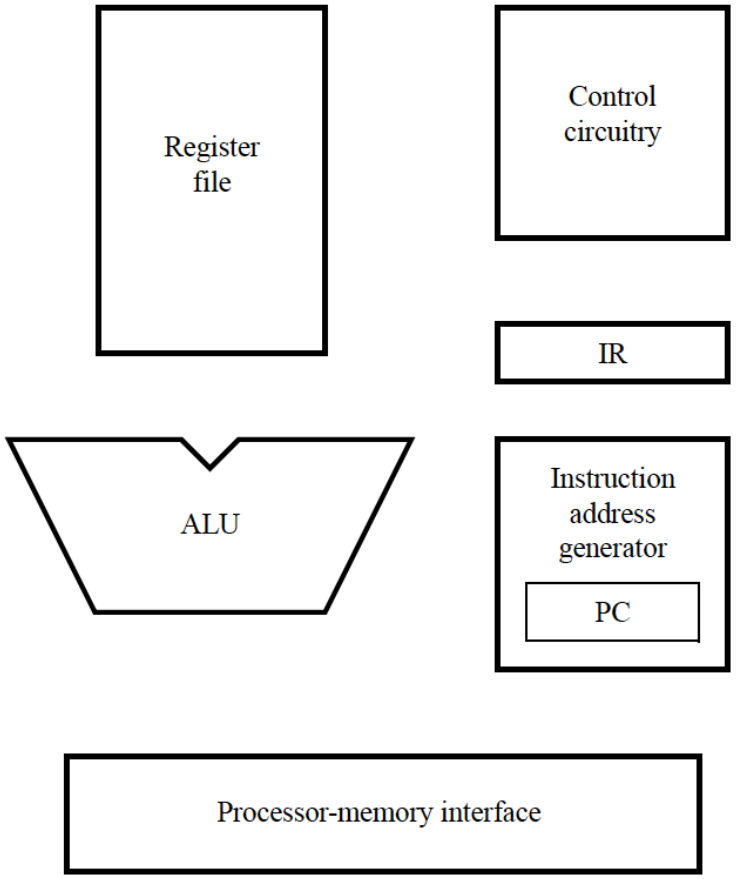


CPU + memory Abstraction



Processor's main building blocks

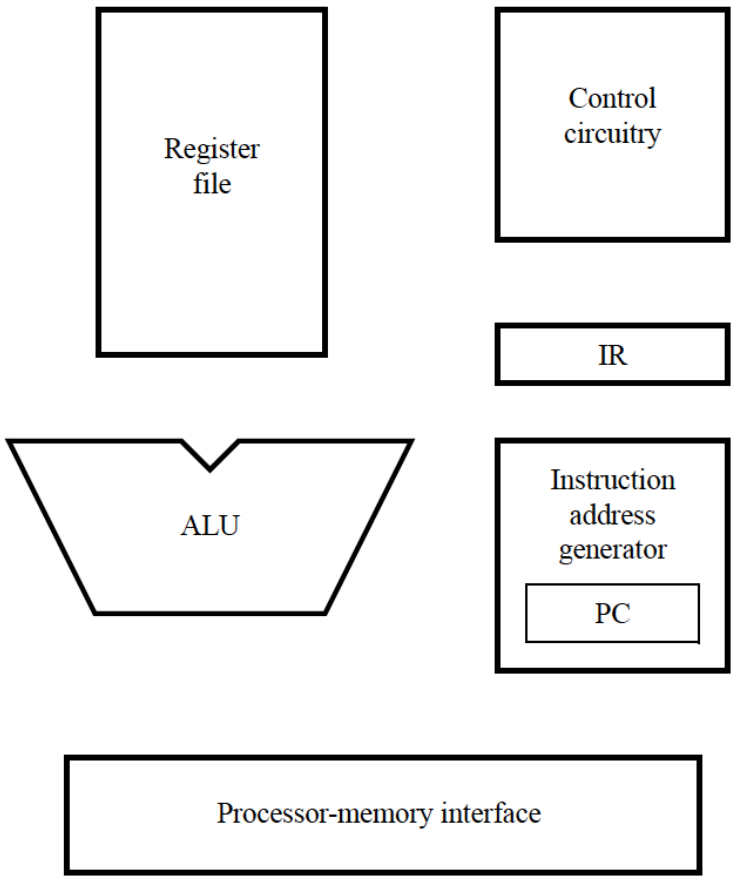
- PC provides instruction address.
- Instruction is fetched into IR
- Instruction address generator updates PC
- Control circuitry interprets instruction and generates control signals to perform the actions needed.



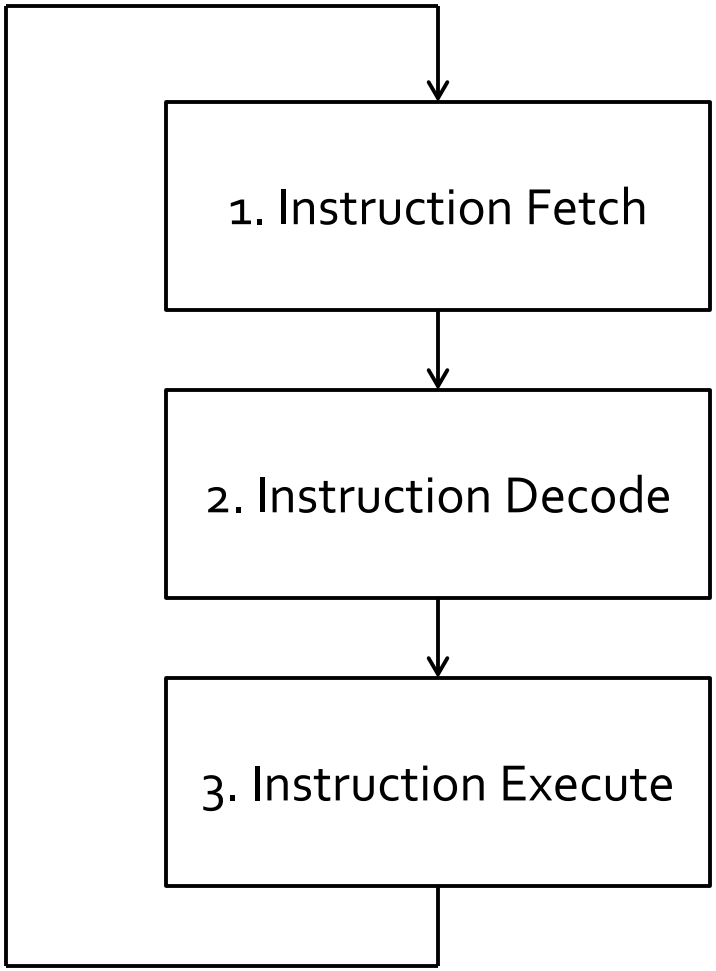
Datapath and controlpath examples for different instruction types.

Processor's main building blocks

```
MAX:    Subtract    SP, SP, #20
        Store       R1, 16(SP)          Push R1
        Store       R2, 12(SP)          Push R2
                    …                   …
        Store       R5, (SP)            Push R5
        Load        R3, 24(SP)
        Load        R5, 20(SP)
        Move        R2, #1
Loop:   Add         R5, R5, #4
        Load        R4, (R5)
        Branch_if_(R1>=R4) Skip
        Move        R1, R4
Skip:   Add         R2, R2, #1
        Branch_if_(R3>R2) Loop
        Store       R1, 24(SP)
        Load        R1, 16(SP)          Restore R1
        Load        R2, 12(SP)          Restore R2
                    …                   …
        Load        R5, (SP)            Restore R5
        Return
```



Processor Loop



Instruction Fetch

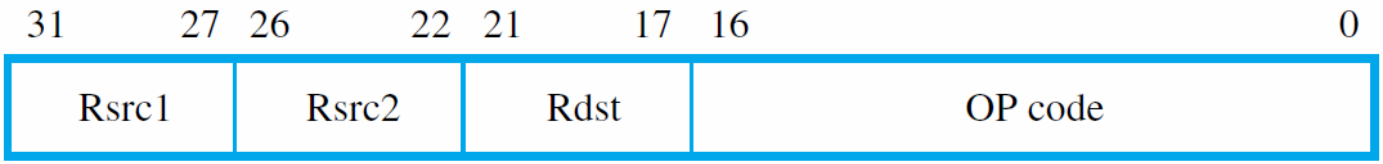
- Common to all instructions
- (Essential) Read instruction word in memory pointed to by PC into IR
- (Optional) Increment PC (may be overwritten if the next instruction is different)
  - **Important architectural principle:** *Make the common case fast.*

Instruction Decode

- Common to all instructions
- (Essential) Figure out what the instruction is so that the control signals can be correctly generated for instruction execution.
  - Done by decoding the instruction opcode field(s)
- (Optional) Read *source registers* specified in by the register fields of instructions involving register operands.
  - Wasted effort if the instruction does not involve register reads but this is the common case.

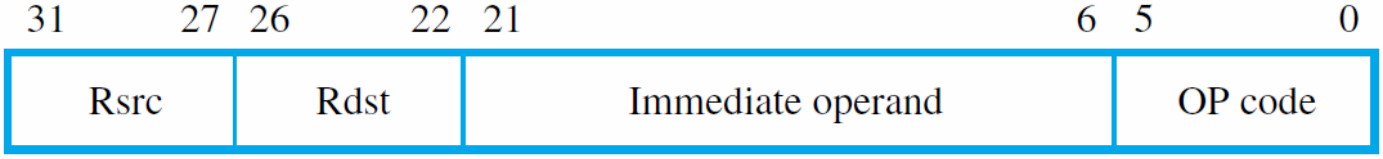
Instruction Decode

Add R3, R4, R5



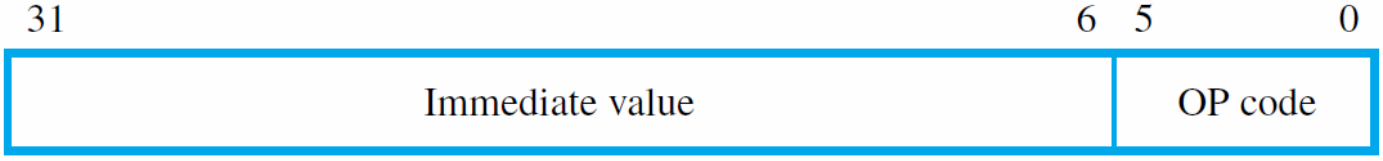
(a) Register-operand format

Load R5, X(R7)



(b) Immediate-operand format

Branch LOOP



(c) Call format

Instruction Execution

- Depends on *instruction type.*

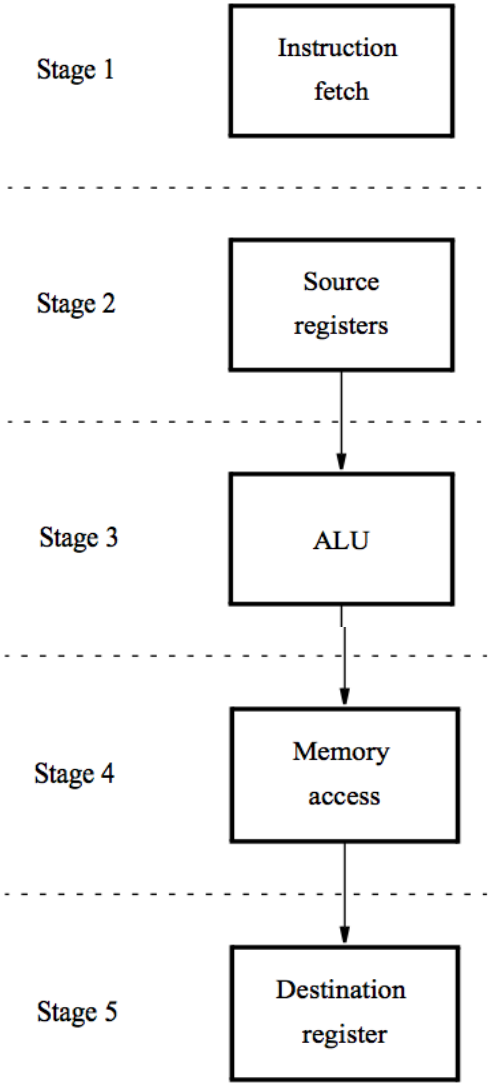
- *Can be further divided into stages*

Instruction Execution

- Load R5, X(R7)
- Add R3, R4, R5
- Add R3, R4, #100
- Store R6, X(R8)

Five-Stage Execution

- Instruction processing moves from each stage to the next in every clock cycle, after fetch in Stage 1.
- The instruction is decoded and the source registers are read in stage 2.
- Computation takes place in the ALU in stage 3.
- If a memory operation is involved, it takes place in stage 4.
- The result of the instruction is stored in the destination register in stage 5.



Datapath: Stages 2–5

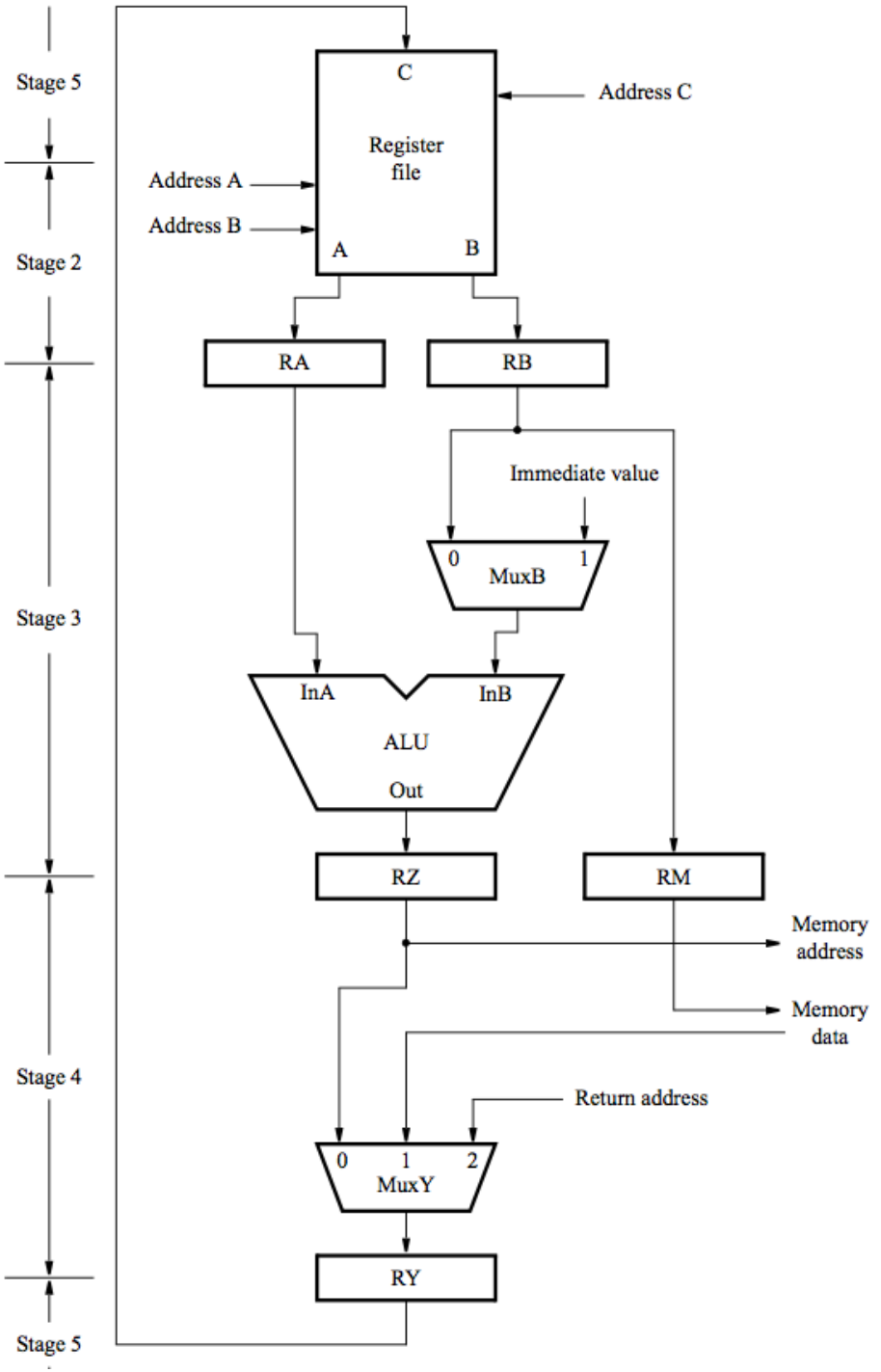
2. Register

Stage 1 (Instruction Fetch) is shown later.

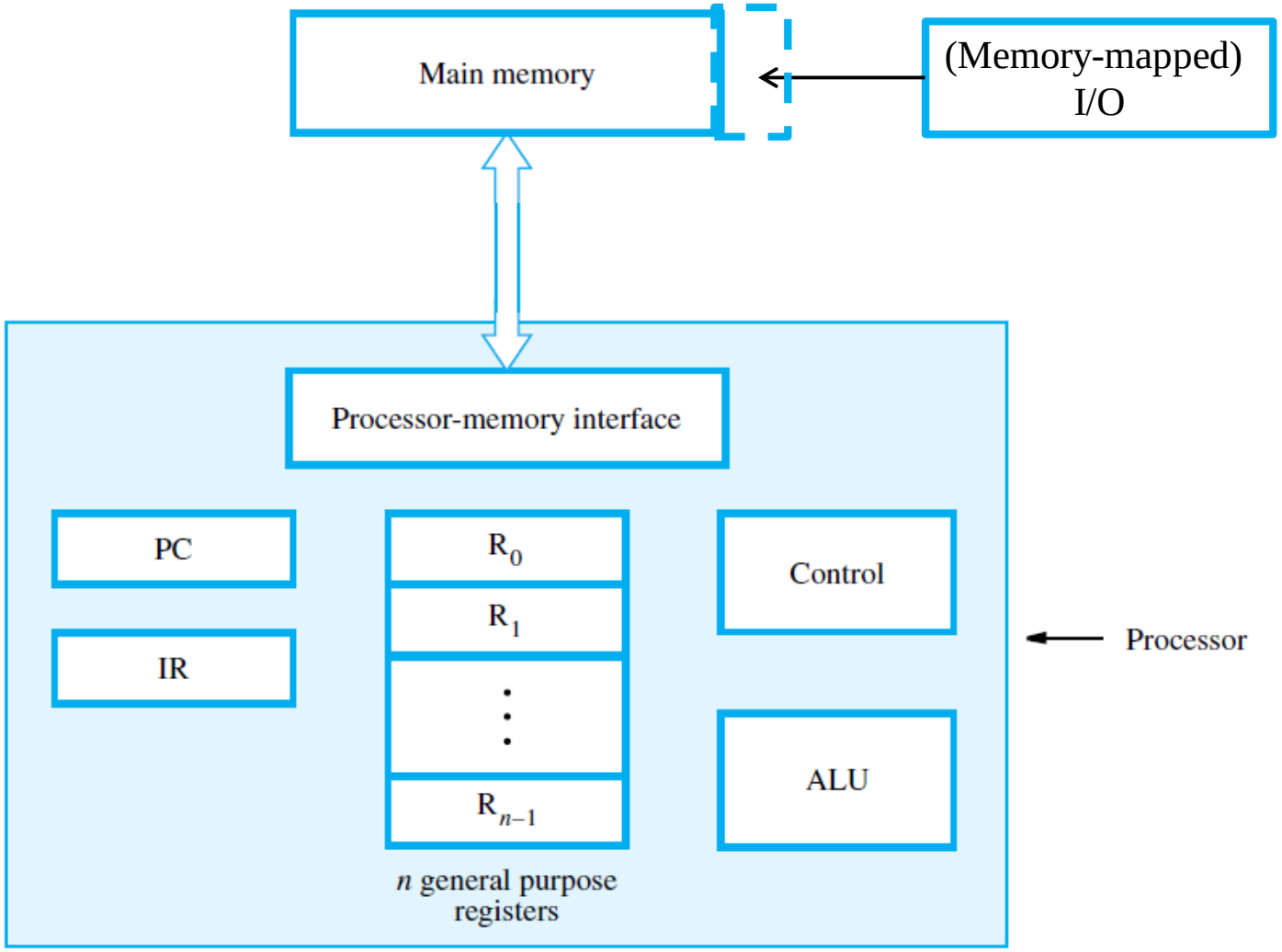
3. ALU

4. Memory

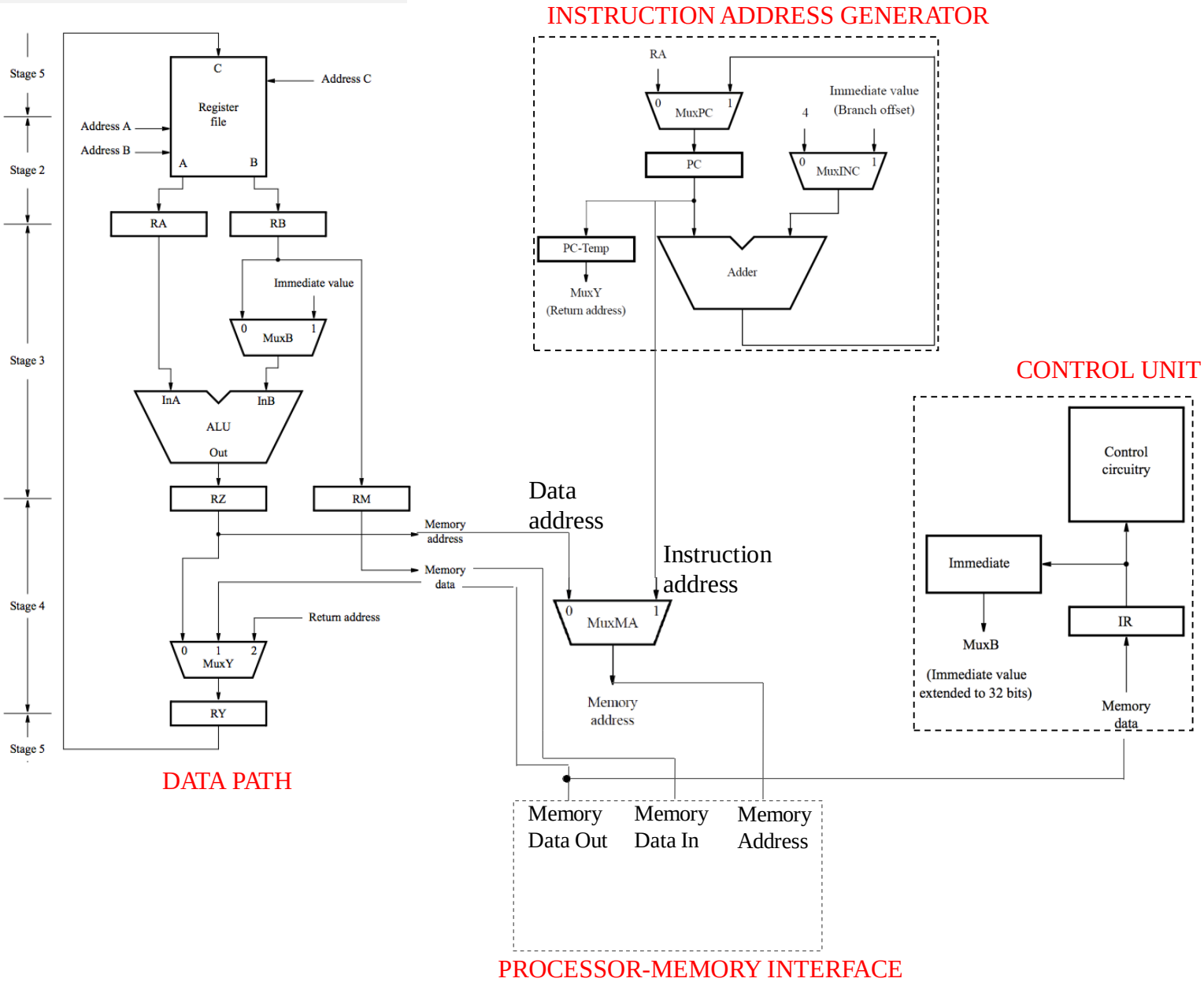
5. Writeback



Computer Architecture

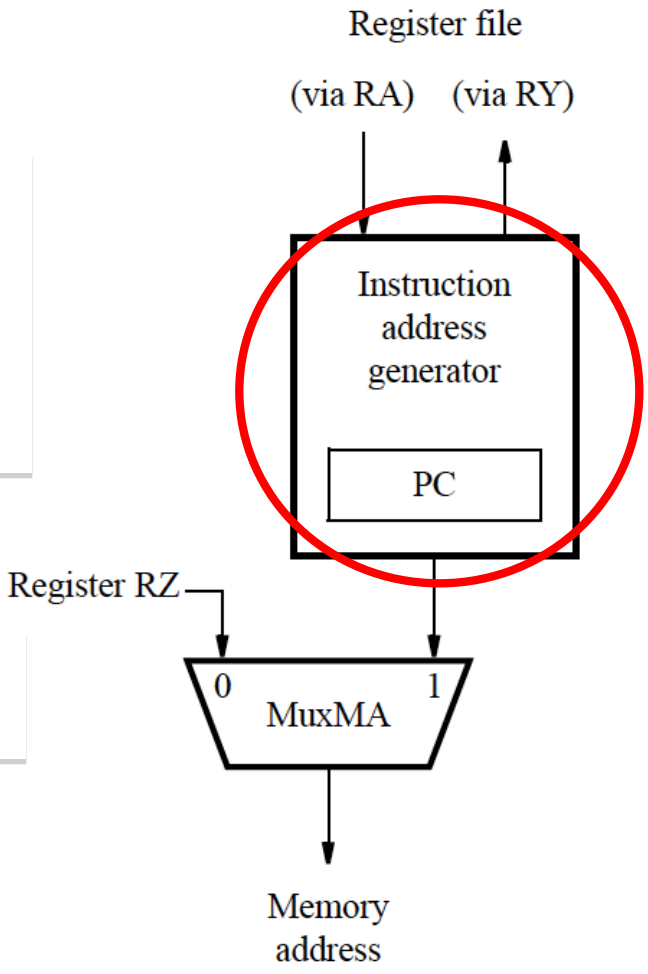


Complete Datapath



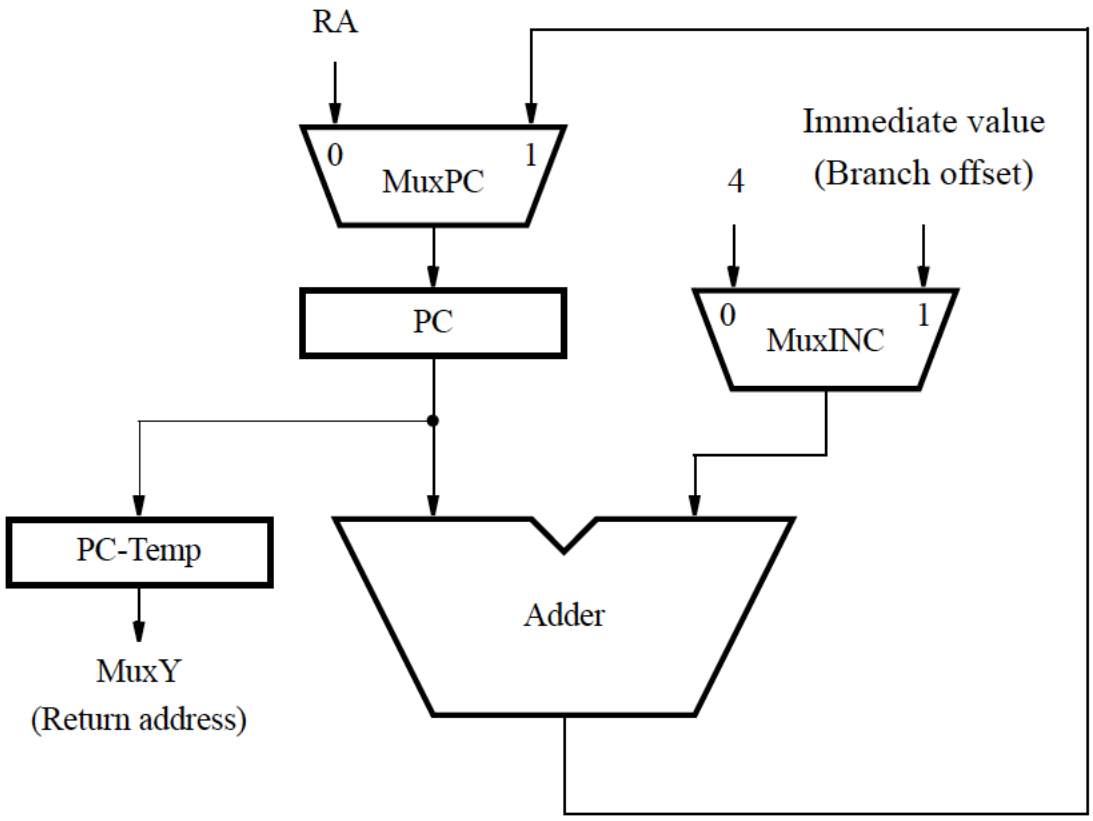
Memory address generation

- Two kinds:
  - Instruction address (via Input 1)
  - Data address (Via Input 0)
- Instruction address generator updates PC after instruction fetch.
- Also generates (subroutine) branch and subroutine return addresses.
- MuxMA selects RZ when reading or writing data operands.
  - RZ stores the address computed by the ALU.



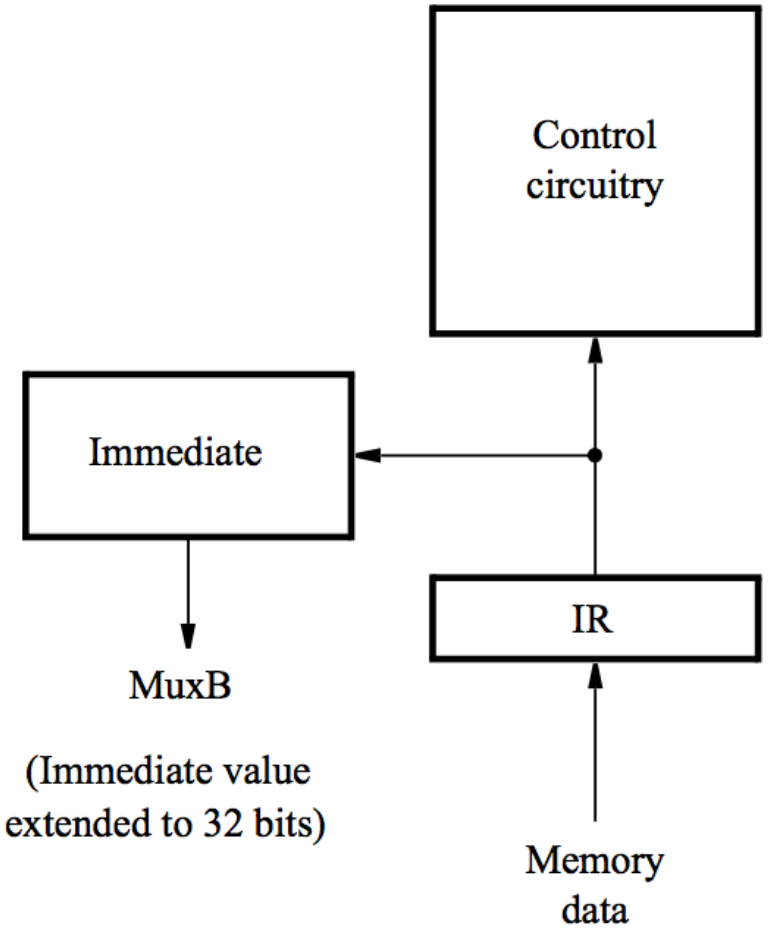
Instruction address generator

- Connections to RA and RY (via MuxY) are used for subroutine call and return.



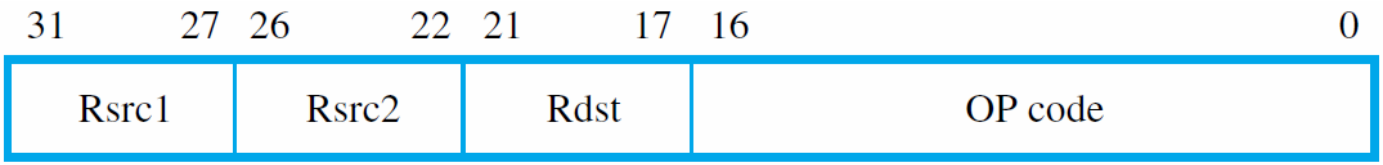
Processor control section

- When an instruction is read, it is placed in IR.
- The control circuitry decodes the instruction and generates the control signals that drive all units (more on this later).
- The Immediate block extends the immediate operand to 32 bits as specified in the instruction.

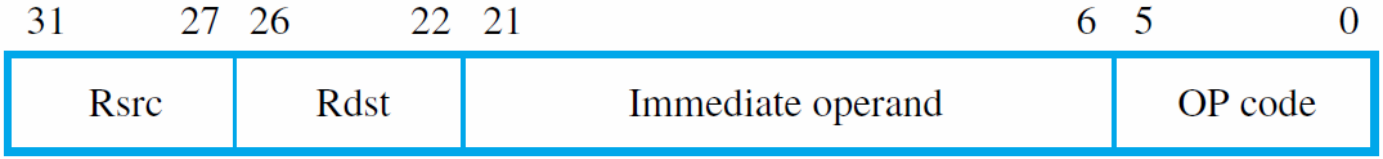


Instruction Formats (Chapter 5 Processor)

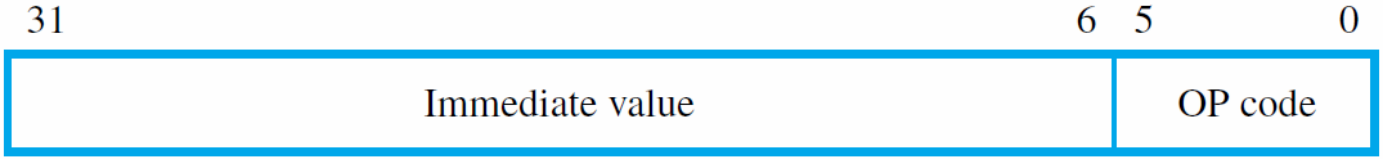
Fig. 5.12



(a) Register-operand format



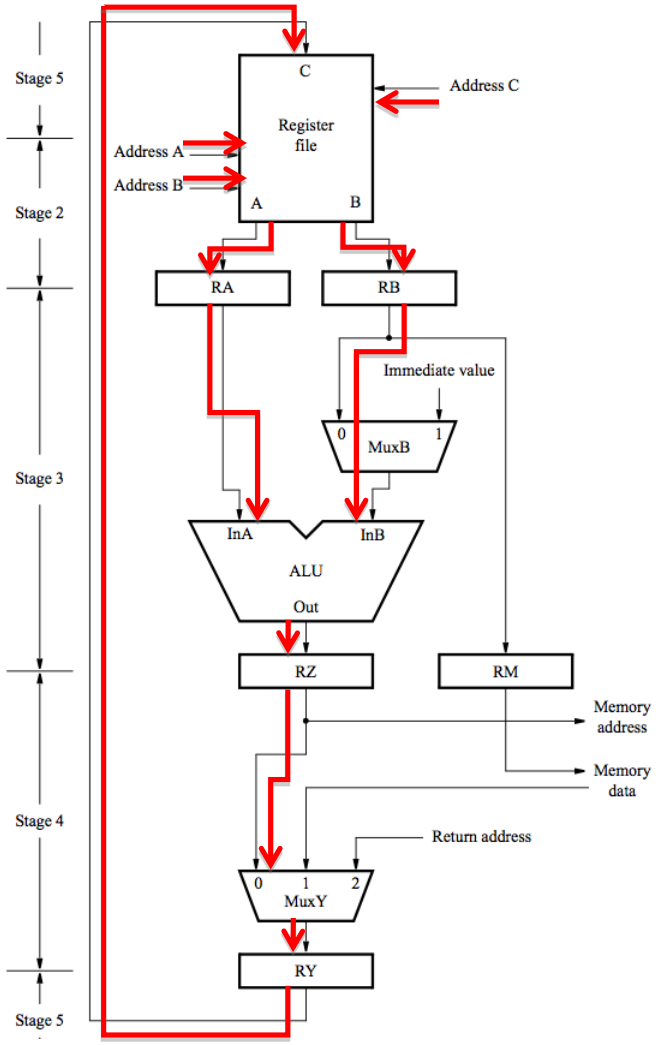
(b) Immediate-operand format



(c) Call format

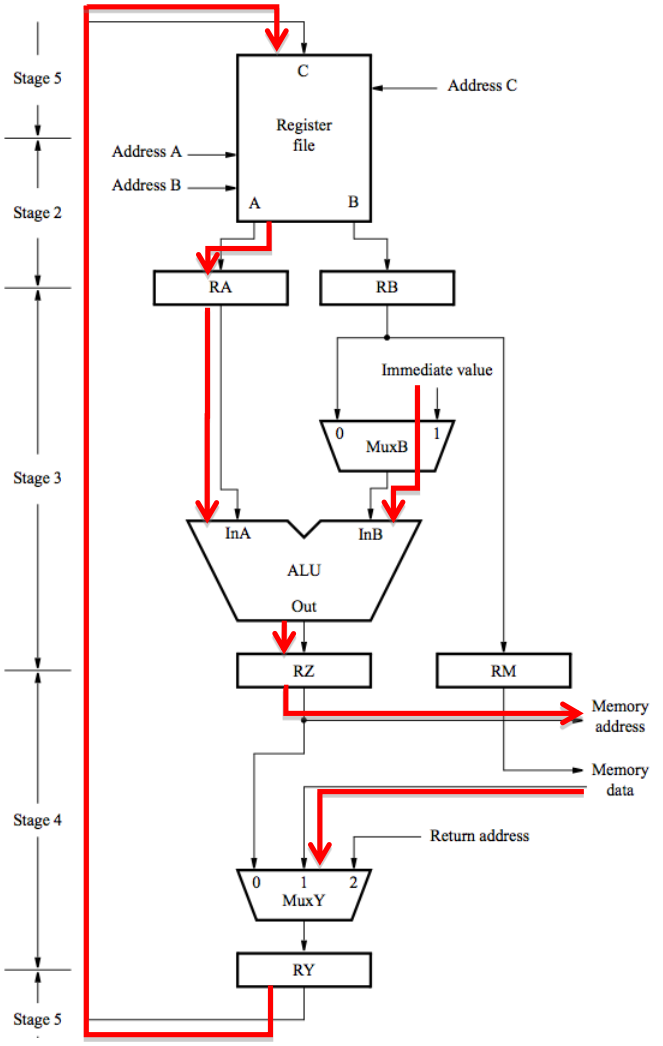
Example: Add R3, R4, R5

1. Memory address ← [PC],
   Read memory,
   IR ←Memory data,
   PC ← [PC] + 4
2. Decode instruction,
   RA ← [R4], RB ← [R5]
3. RZ ← [RA] + [RB]
4. RY ← [RZ]
5. R3 ← [RY]



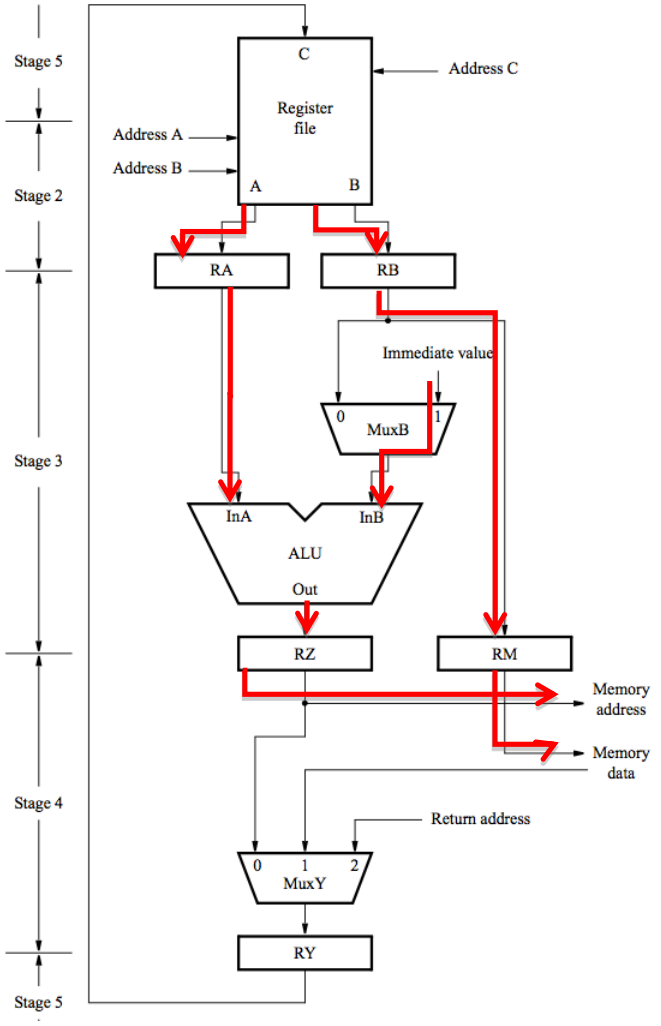
Example: Load R5, X(R7)

1. Memory address ← [PC],
   Read memory,
   IR ← Memory data,
   PC ←[PC] + 4
2. Decode instruction, RA ←[R7]
3. RZ ←[RA] + Immediate value X
4. Memory address ←[RZ],
   Read memory,
   RY ← Memory data
5. R5 ←[RY]



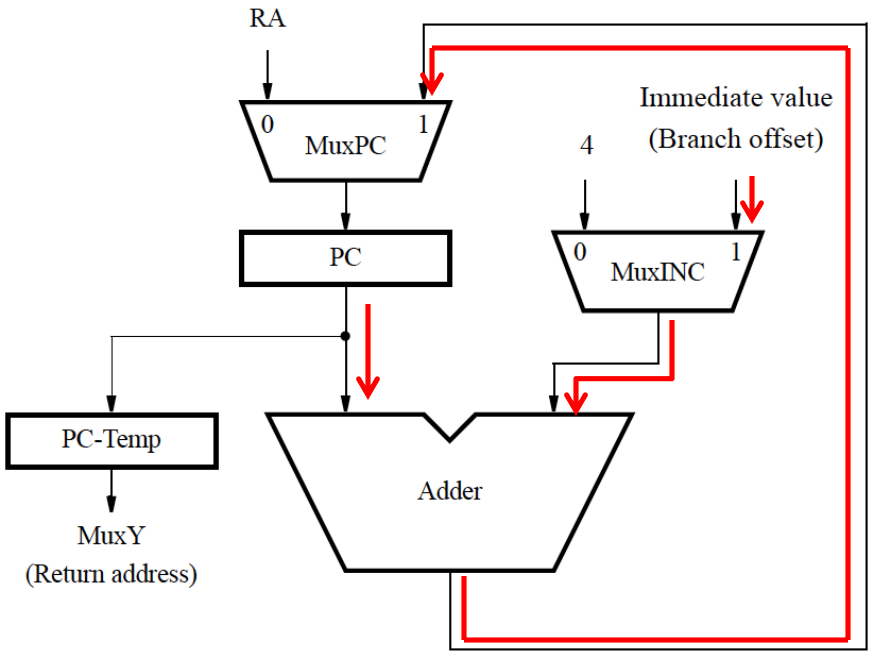
Example: Store R6, X(R8)

1. Memory address ←[PC],
   Read memory,
   IR ← Memory data,
   PC ← [PC] + 4
2. Decode instruction,
   RA ←[R8], RB ←[R6]
3. RZ ←[RA] + Immediate value X,
   RM ←[RB]
4. Memory address ←[RZ],
   Memory data ←[RM],
   Write memory
5. No action



Unconditional branch

1. Memory address ←[PC]
   Read memory,
   IR ← Memory data,
   PC ←[PC] + 4
2. Decode instruction
3. PC ←[PC] + Branch offset
4. No action
5. No action



Conditional branch: Branch_if_[R5]=[R6] LOOP

1. Memory address ←[PC],
   Read memory,
   IR ← Memory data,
   PC ←[PC] + 4
2. Decode instruction, RA ←[R5], RB ←[R6]
3. Compare [RA] to [RB],
   If [RA] = [RB], then PC ←[PC] + Branch offset
4. No action
5. No action

Complete the data flow diagram on your own.

Subroutine call with indirection: Call_register R9

1. Memory address ←[PC],
   Read memory, IR ← Memory data,
   PC ←[PC] + 4
2. Decode instruction, RA ←[R9]
3. PC-Temp ←[PC],
   PC ←[RA]
4. RY ← [PC-Temp]
5. Register LINK ← [RY]

Complete the data flow diagram on your own.

Course Project

- 20% of your course grade
- Goals
  - Understand software/hardware interface better by implementing a substantial subset of a Reduced Instruction Set Computer (RISC) instruction set architecture (ISA).
  - Understand design parameters that determine the performance of hardware design in terms of timing and utilization of hardware resources.
  - Learn to work as a team to carry out a complex design task requiring task partitioning, effective communication, and cooperation.
  - Produce a report that accurately describes your work according to the best practice in technical communication.

Course Project

- Implement a basic processor with
  - an ISA that strongly resembles a subset of the MIPS architecture
  - some features that are unique to ARM
  - all instructions are 24-bits wide and data is 16-bits wide.
  - pipelining
  - (optional) interrupt capabilities

Project Timeline

- Consists of 7 steps
- Step 7 – Optional enhancements = Extra credit

- 1- Oct. 29/31 : ALU and Register File (done in lab)
- 2- Nov. 5/7 : Connect ALU and Reg File to datapath & mini report
- 3- Nov. 12/14 : Implement remaining instructions & mini report
- 4- Nov. 19/21 : Add I/O and ARM-like conditional execution & mini report
- 5- Dec. 3/5 : Add pipelining & mini report
- 6- Dec 10/12: Ideally have project done (including extra credit), report started.
- 7- Dec 19 : (Finals week) Present (actually need everything done): Project, Report, Demonstration

- One week before each due date, a handout will be given with more detail on what is due.

Project Reporting

- Every week you will submit a three-page report describing the portion of the project you worked on that week.
- Concluded with a final report
- More information → Course Documents/ Project section of Blackboard.

Project Grading

- 500 points → 20% of course grade (+150)
- Parts 2-4: 50 Pts each
- Part 5: 100 Pts
- Final Part: 200 Pts
  - 100 Pts: Report
  - 50 Pts: Presentation
  - 50 Pts: Demonstration
- Individual: 50 Pts
  - Our assessment of each member's contribution (5 Pts/week)
  - Peer evaluation form (25 Pts)
- Bonus: Up to 150 Pts – Depends on the difficulty and success of the extras

Upcoming…

- Homework 4 – Chapter A
  - Due Friday, Oct. 25th
- Quiz 4 – Chapter A (A.1-A.5)
  - Monday, Oct. 28th (15 min)
- Midterm – Chapter 1, 2, 3
  - Monday, Nov. 4th (50 min)
- Test 2 – Chapters A & 5
  - Monday, Nov. 11th (50 min)