The Edward S. Rogers Sr. Department of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

ECE342 – Computer Hardware
Lecture 21

Bruno Korst, P.Eng.

Agenda

- Processing Unit
  - Measures of performance
  - Instruction processing and CPU components

Processing Unit

- Consider two implementations of the same program using the same ISA (Instruction Set Architecture)
  - Computer A, clock cycle = 250ps, CPI of 2.0
  - Computer B, clock cycle = 500ps, CPI of 1.2
- At first glance, A → 4GHz, and B→ 2GHz (so… A is faster)
- If **I** is the number of instructions for that one program being run, I know that it should be the same for both
  - but I also know that the CPI is not the same.

Processing Unit

- I can calculate my CPU (execution) time in terms of I

For Computer A, CPUtime = I x 2 x 250 = 500 I

For Computer B, CPUtime = I x 1.2 x 500 = 600 I

- Execution time for B = 1.2 x Execution time for A

- A performs better than B
  - Smaller execution time, better performance

Processing Unit

- It was mentioned earlier that using time alone as a performance metric could be misleading

- The alternative is to use MIPS
  - This is a measure of program execution speed based on “millions of instructions per second”
  - The idea is: faster computers → greater MIPS metric

$$\text{MIPS} = \text{Instruction Count} / (\text{Execution time} \times 10^6)$$

Processing Unit

- Expanding, we have

$$\text{MIPS} = \frac{\text{Instruction Count}}{\dfrac{\text{Instruction Count x CPI}}{\text{Clock rate}} \text{ x } 10\text{^}6} = \text{Clock rate / (CPI x } 10\text{^}6)$$

- Exercise

	Computer A	Computer B
Instruction Count	10 billion	8 billion
Clock freq	4GHz	4GHz
CPI	1.0	1.1

  - MIPS(A) greater

Processing Unit

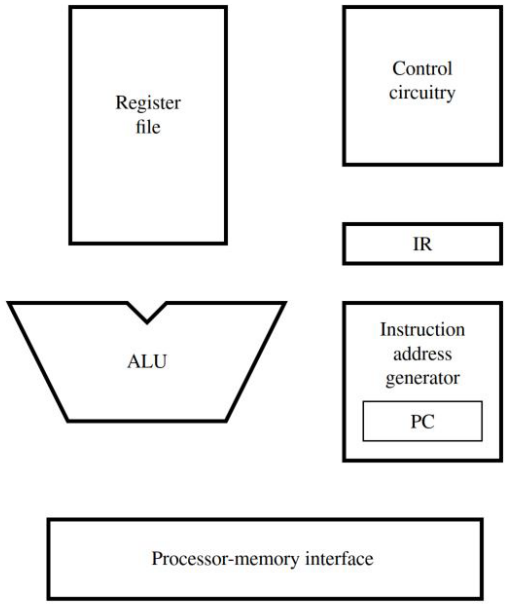
- The Processor
  - (243) “The processor can perform whatever task is needed, by executing programs. It can read data from an external device, write data to a device, compute calculation”
    - We saw that it also participates in the management of devices and the bus.
  - Two main components:
    - Datapath – involves registers holding data currently in use, and ALU for calculation some muxes and interconnect
    - Control – FSM that controls transfers of data in/out of the processor (via registers), data transfers between registers and control of the ALU in each clock cycle

Processing Unit

- We wish now to understand the different stages of instruction processing
- We will describe the functionality of different parts, put them together and describe how the CPU works for different instructions
- For now, they are
  - Register File
  - ALU
  - Datapath (how they connect)
    - Under the fetch stage
    - Under the execution stage

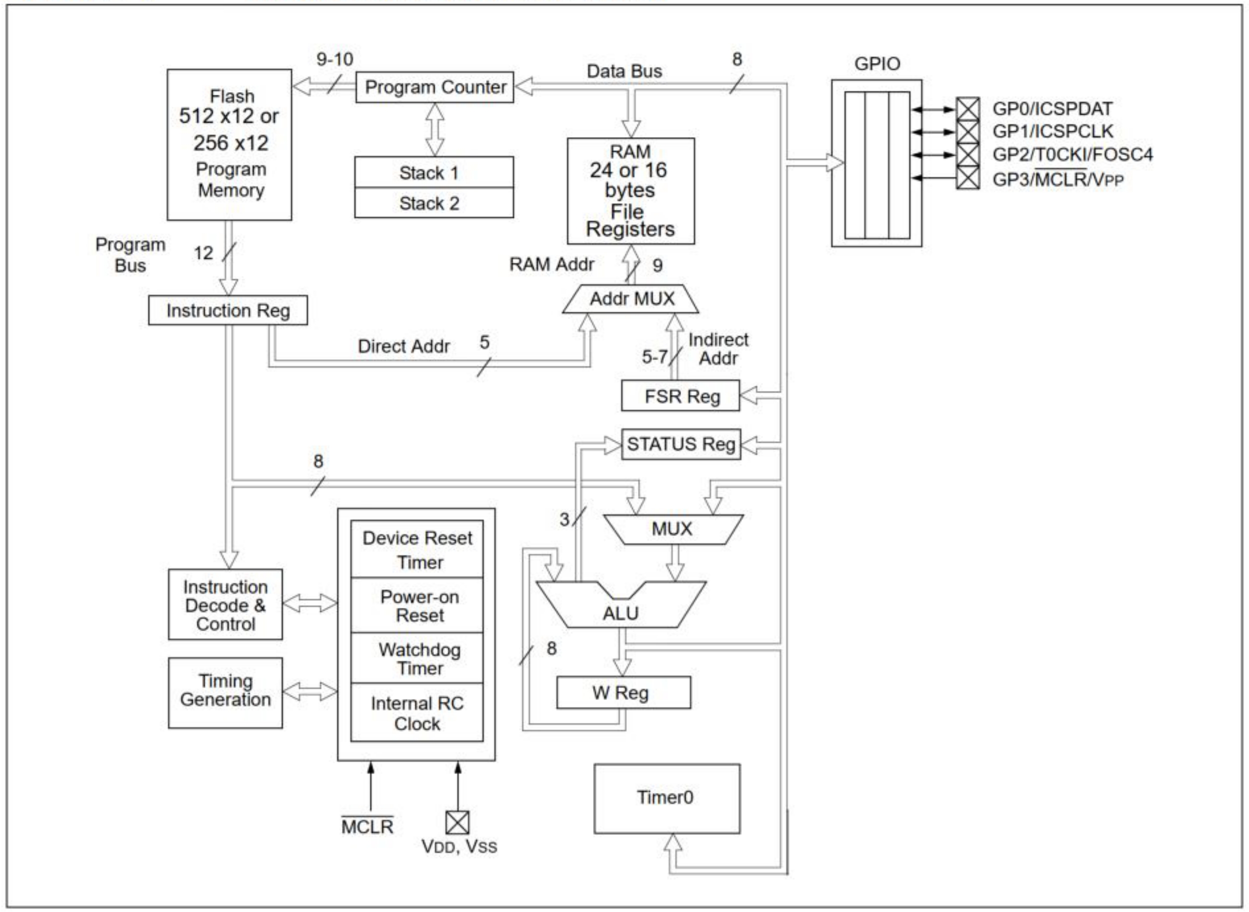
Processing Unit

Simplified version



Real version, but relatively simple

FIGURE 3-1: PIC10F200/202 BLOCK DIAGRAM



Processing Unit

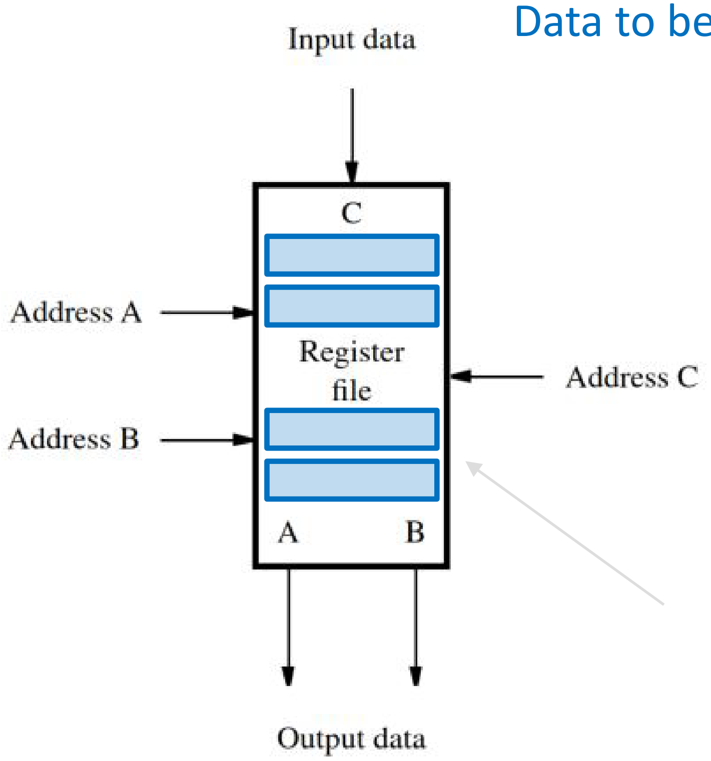
- Register File
  - Small and fast memory block
  - Array of storage elements with access circuitry to read/write
  - Circuitry designed to read 2 at a same time
    - 2 address inputs connected to Instruction Register (IR – instruction contains addresses)
      - These are sources – where to read
    - 2 other inputs: one data input and one address input
      - Destination – where to write

Processing Unit

- Register File

Data to be written

Read from registers @ address…



Address of register to be written to

Number of registers depends on individual CPU design

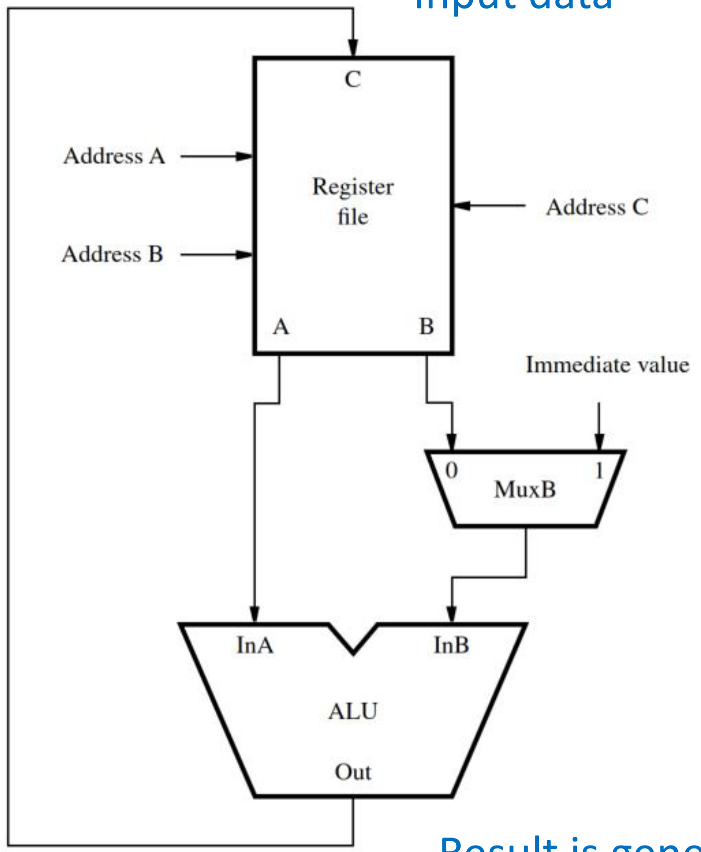
Put out what is read

Processing Unit

- ALU (Arithmetic and Logic Unit)
  - It is used to manipulate data
    - Arithmetic operations (ADD/SUB)
    - Logic operations (AND/OR/XOR)
  - When an instruction is executed
    - Contents of the two registers (in the instruction) are read from the register file and made available @ outputs
    - MUX may specify a direct value or address offset prior to input to ALU

Processing Unit

- In the context of an operation…



Input data

Destination address
(write result to here)

Read from these addresses

A value from IR
- Address offset
- A direct value to be added

Result is generated

Processing Unit

- Datapath
  - Instruction processing consists of 2 phases
    - Fetch phase
    - Execution phase
  - Fetch
    - “Fetch & Decode” the instruction
    - Generate control signals that result in appropriate actions during the execution phase
  - Execution
    - Read data operands, perform operation, stores results

Processing Unit

- Let’s look at a RISC style instruction
  - All follow a 5 step sequence of actions
  - It is one word long
  - Only LOAD and STORE instructions access operands in memory
  - Computations take data from
    - General purpose registers, or…
    - Direct/immediate value coming from the instruction

Processing Unit

- RISC instruction stages
  - 1. Fetch instruction and increment PC
  - 2. Decode instruction and read registers from register file
  - 3. Perform arithmetic/logic operation
  - 4. Read from or write to memory (if needed)
  - 5. Write result in destination register

Later on we'll see IF-ID-EX-MEM-WB

Processing Unit

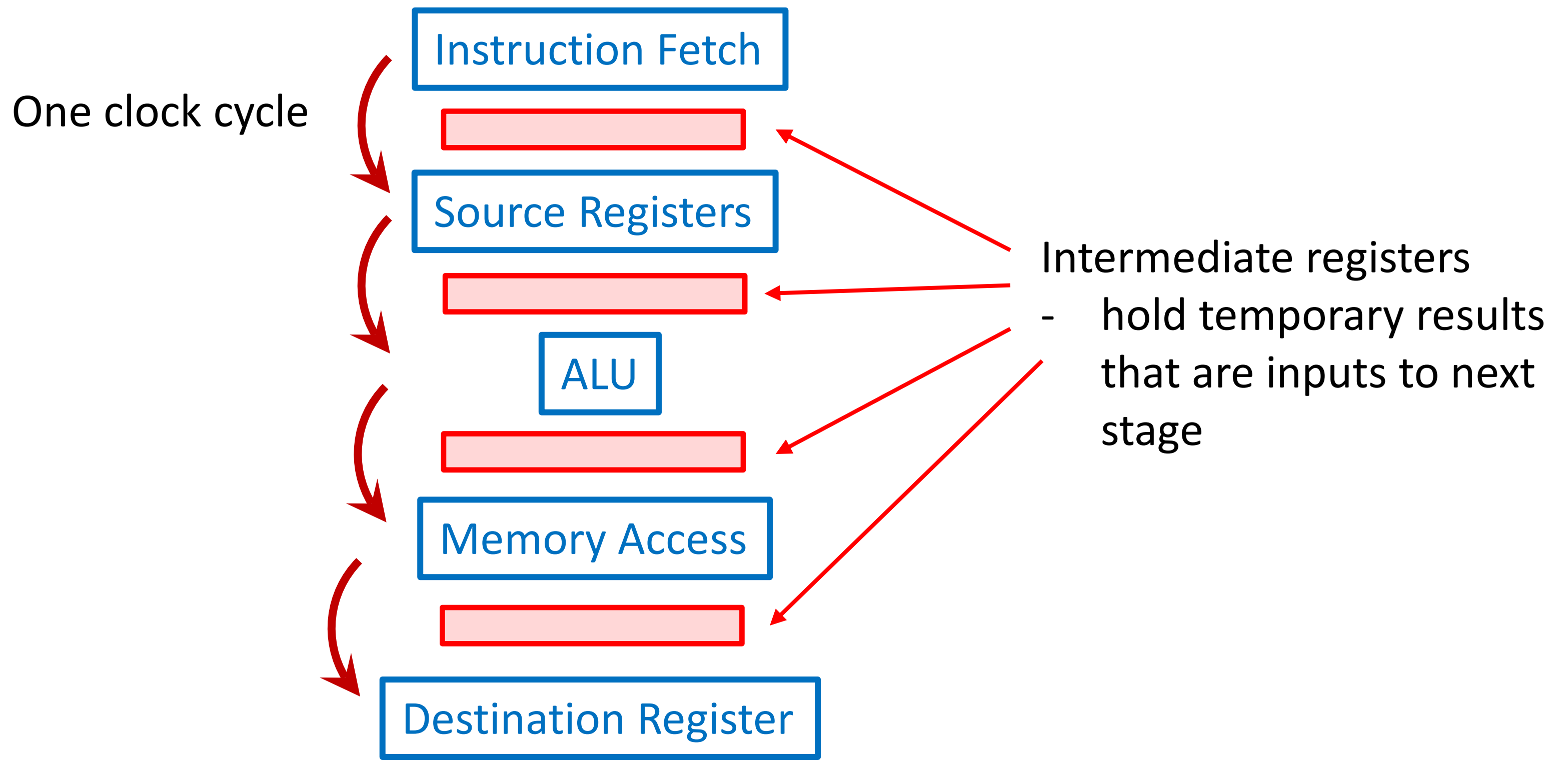
- Example
  - LOAD R5, x(R7)
    - Memory location x + R7 loaded into R5
    - 1) Fetch and increment PC
    - 2) Decode and read R7
    - 3) Compute address (ALU)
    - 4) Read memory
    - 5) Load it into R5

Processing Unit

- Example
  - ADD R3, R4, R5
    - Add contents of R4 to contents of R5, put in R3
    - Note: these are not in memory → no step 4
    - 1) Fetch, increment PC
    - 2) Decode, read R4 and R5
    - 3) Perform addition R4+R5
    - 4) do nothing
    - 5) Load result into R3

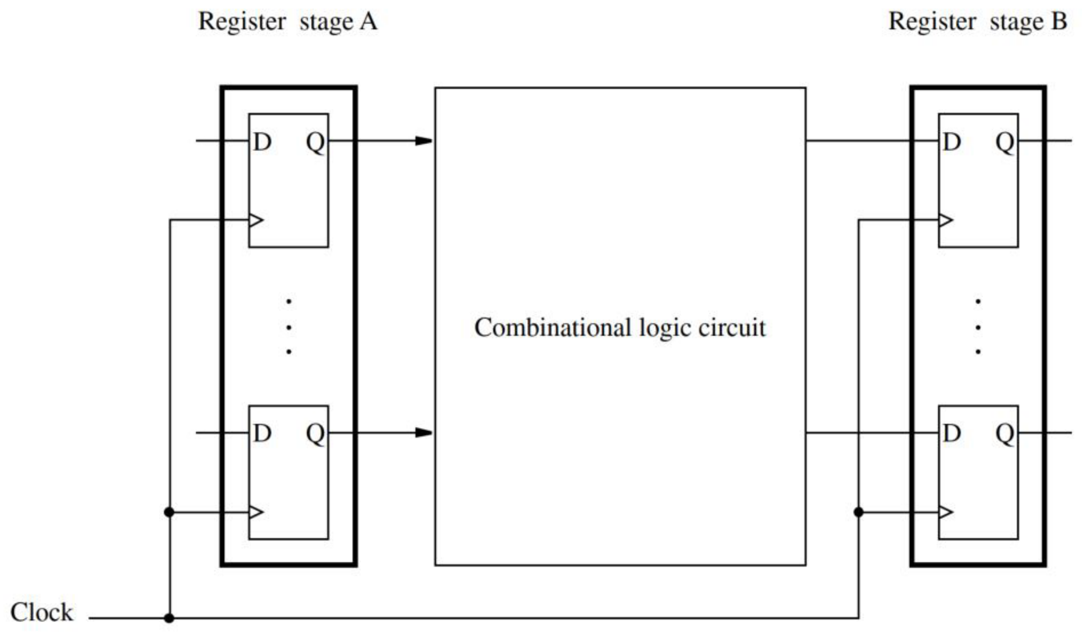
Processing Unit

- If the instructions operate this way, the hardware is organized accordingly



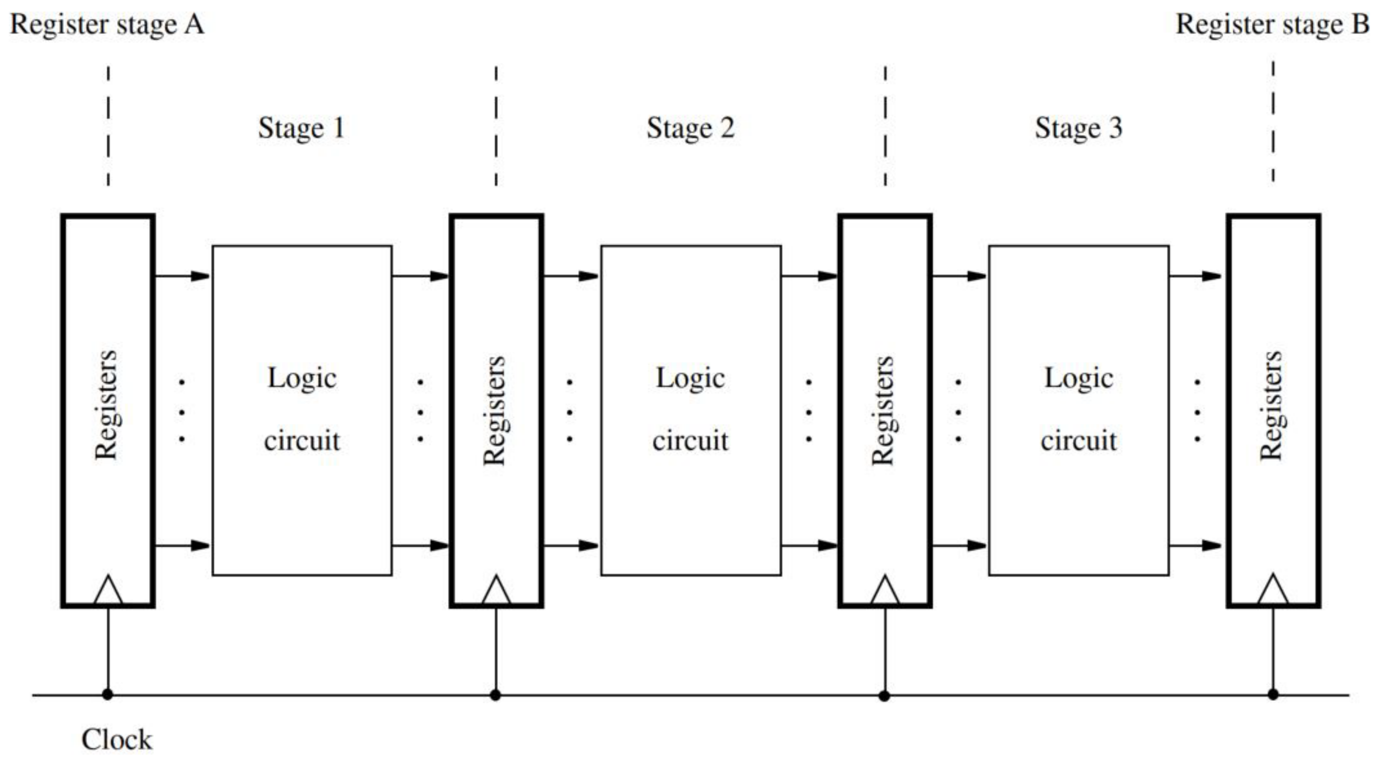
Processing Unit

- Let's pause for a moment
  - "stored" → registers (edge-triggered FF)
  - "processed" → combinational logic
  - "transferred" → clock tick (how fast?)



Processing Unit

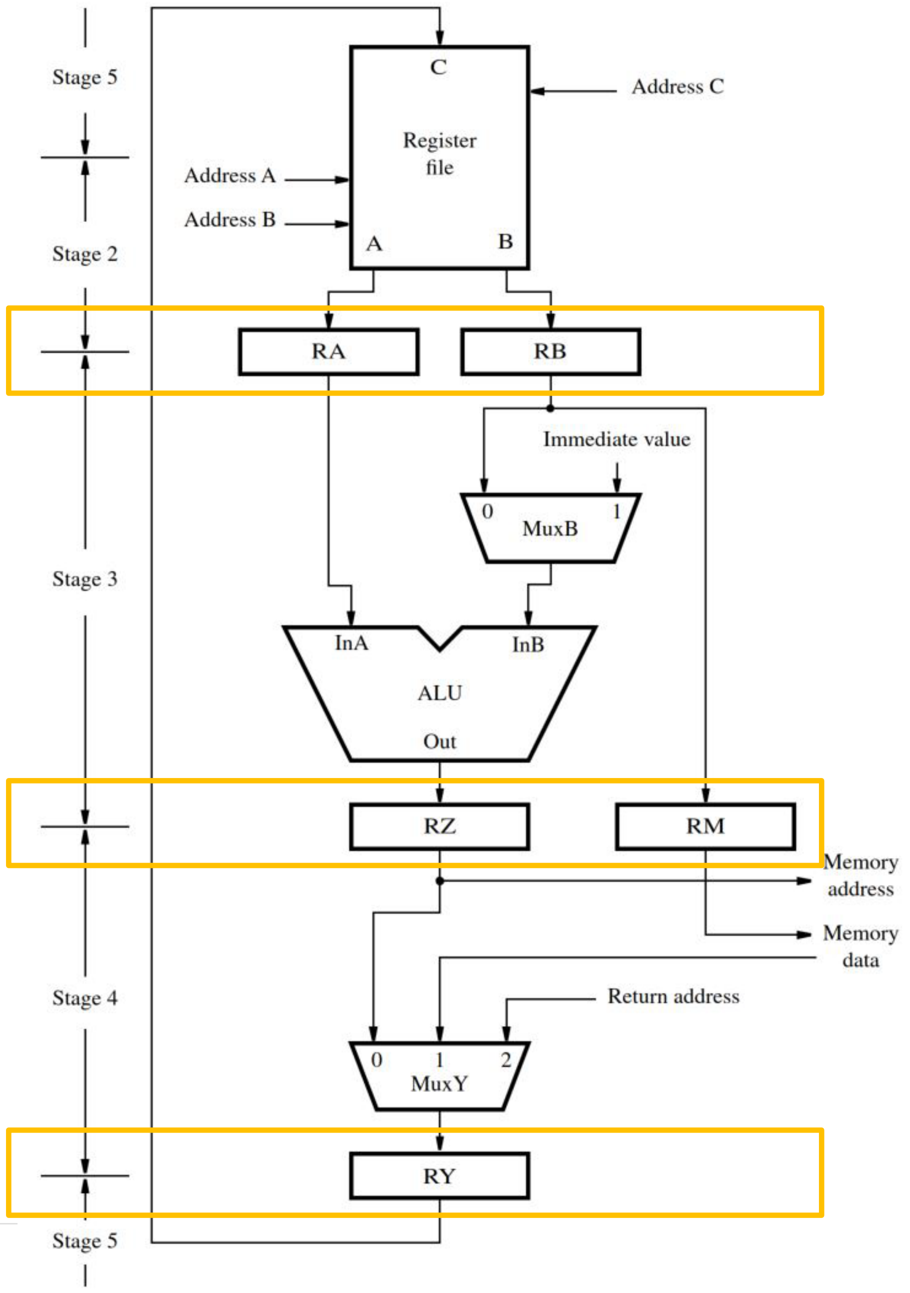
- This process can also be broken down further: stages within stages



Processing Unit

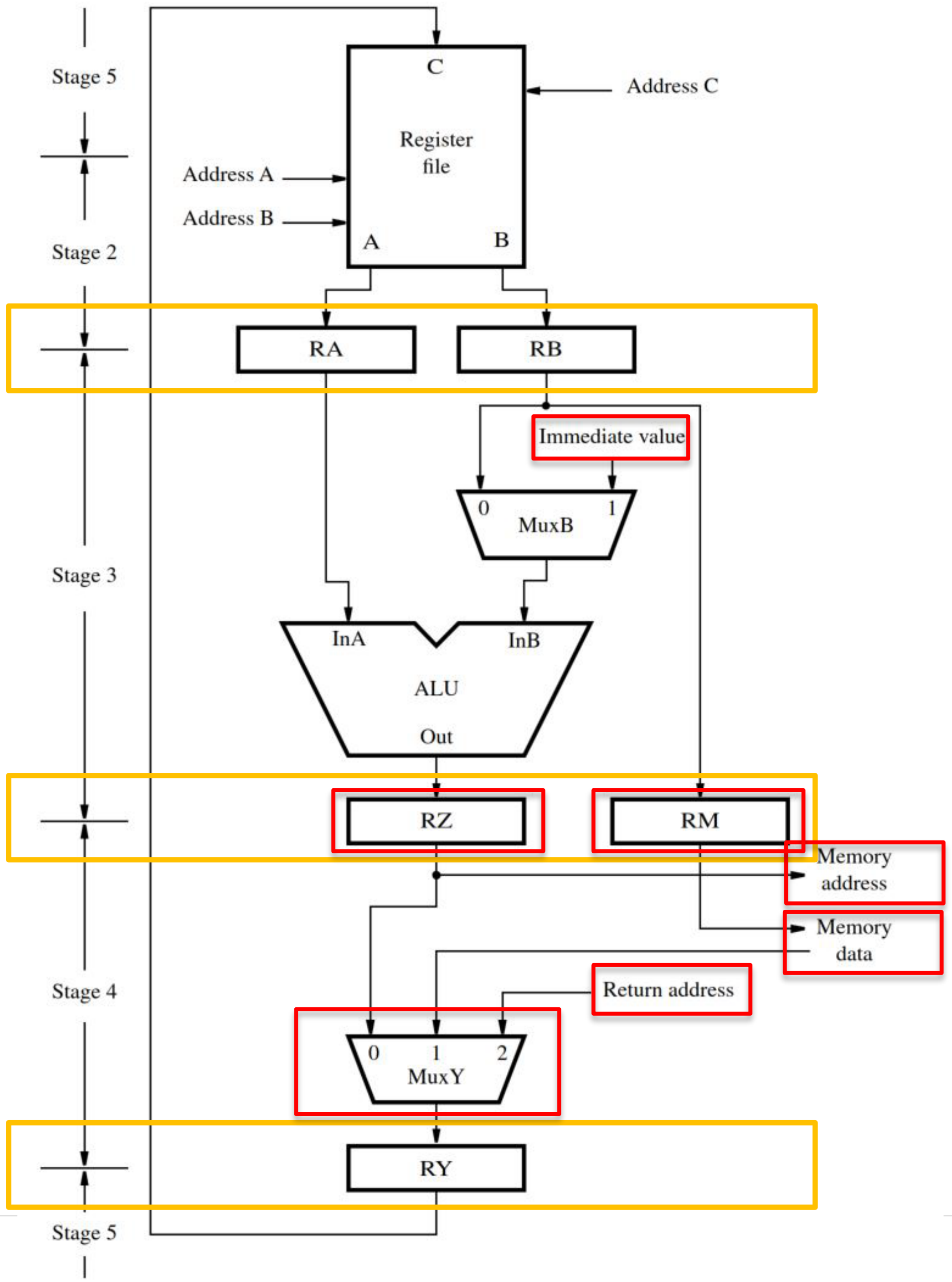
- Adding registers to datapath

Note: fetch/decode has been done (stage 1)
Stage 2 – source registers
Stage 3 – ALU
Stage 4 – Mem access (if needed)
Stage 5 – writes RY into Register File



Processing Unit

- Immediate value
  - Comes from IR
  - It is within the instruction
- RZ can be number or address
  - Sent to MEM or moved on to RY
  - Ex: instruction LOAD or STORE
- RM data to be written to MEM
- Return address
  - Return from subroutine
- MUXY selector examples
  - Instruction ADD – zero
  - Instruction LOAD – one
  - Return from subroutine - two



Processing Unit

- Datapath: Fetch
  - When fetching instructions
    - Address comes from PC (program counter)
    - Instruction is read from memory, placed in IR
      - IR – instruction register
    - Instruction address generator updates PC
      - PC is part of the instruction address generator
    - Instruction stays in IR until execution completed
    - Next instruction is fetched

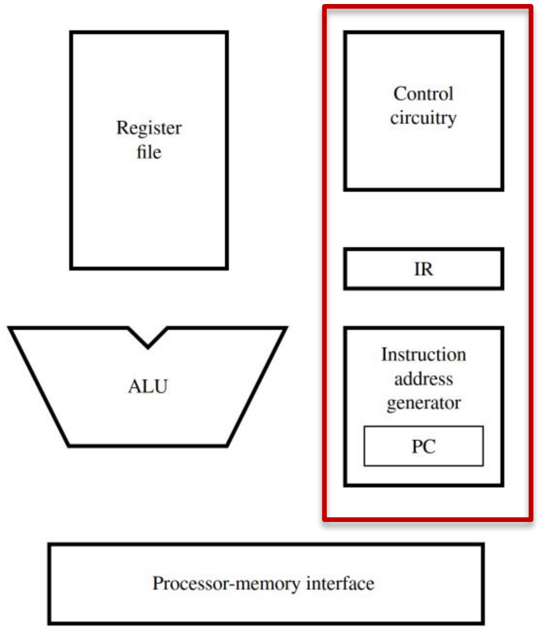
Processing Unit

- Datapath: Fetch (cont'd)
  - When fetching operands
    - Address comes from register RZ
      - MUX selects what goes to processor-memory interface
- Fetch circuitry
  - Involves an instruction address generator (with the program counter)
  - Involves control circuit to generate signals to other hardware in processor

Processing Unit

- Fetch circuitry
  - Involves an instruction address generator (with the program counter)
  - Involves control circuit to generate signals to other hardware in processor

Simplified version



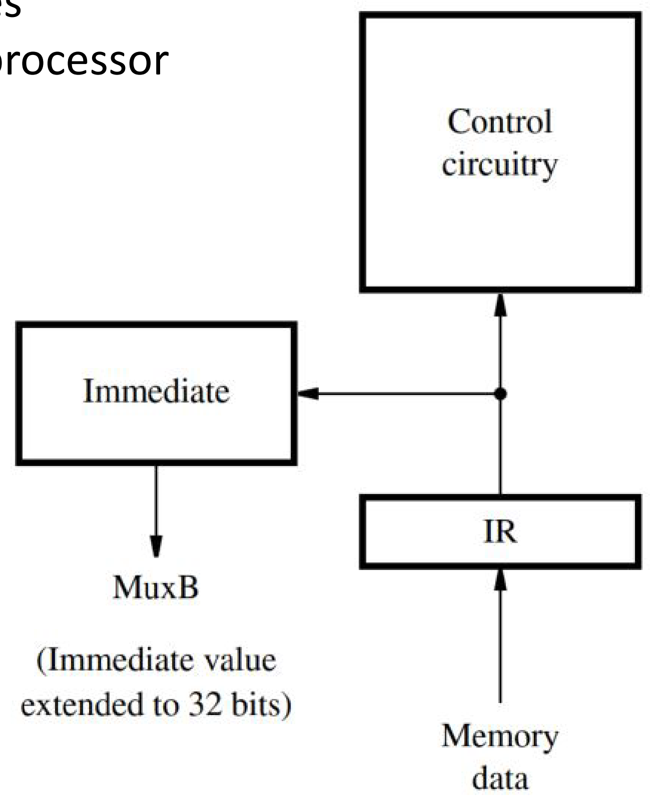
Processing Unit

- Control circuitry OF fetch section

Control circuitry generates signals to other parts of processor based on instruction (IR)

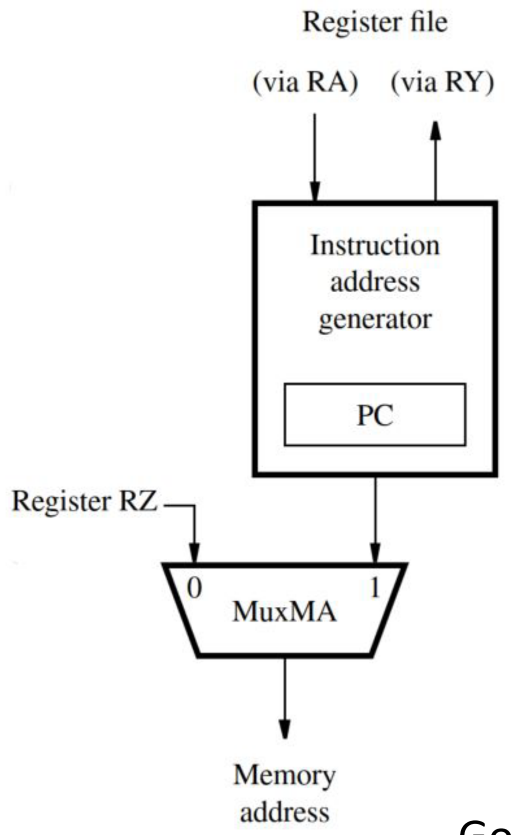
Contents of IR may be used as immediate value
- Fed into MuxB
- Ex:
    - sign extension (arith)
    - Zero padding (logic)
    - Offset for branching

Instruction comes FROM Memory, into IR
- Stays there until end of execution



Processing Unit

- Instruction fetch section



When address is returned from memory, goes via MuxY to RY
- Select (2)

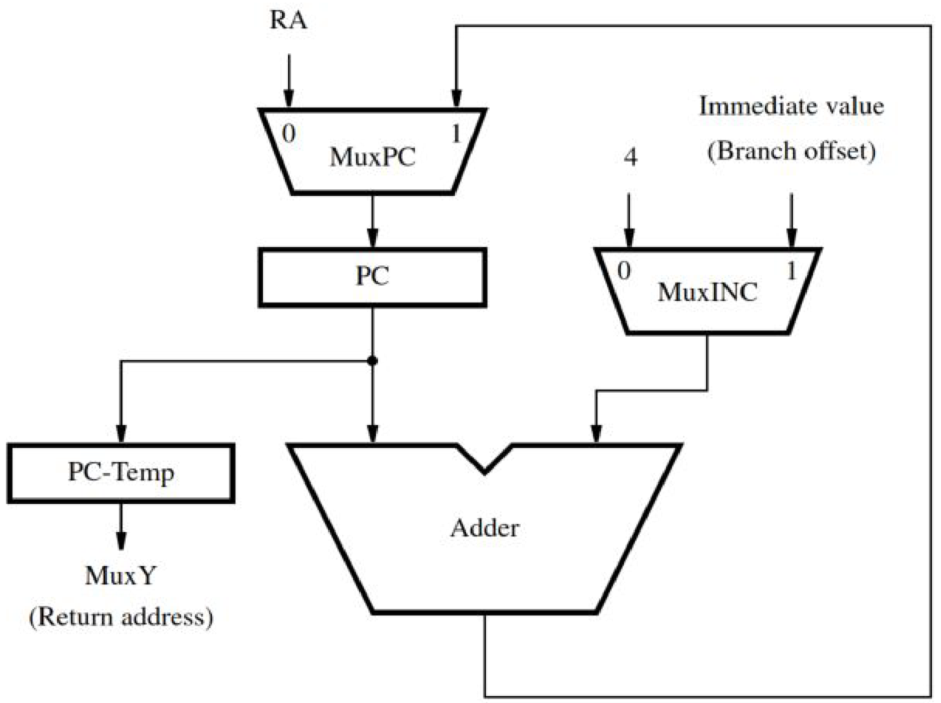
MuxMA selects
- Instruction (1)
- Operand (0)

Address from which to read comes from
- PC when fetching instructions
- RZ when fetching operands

Goes to processor-memory interface

Processing Unit

- Going further into the Address Generator box



MuxINC
- Increments PC by 4 (next address)
- Increments PC by branch offset (subroutine call)

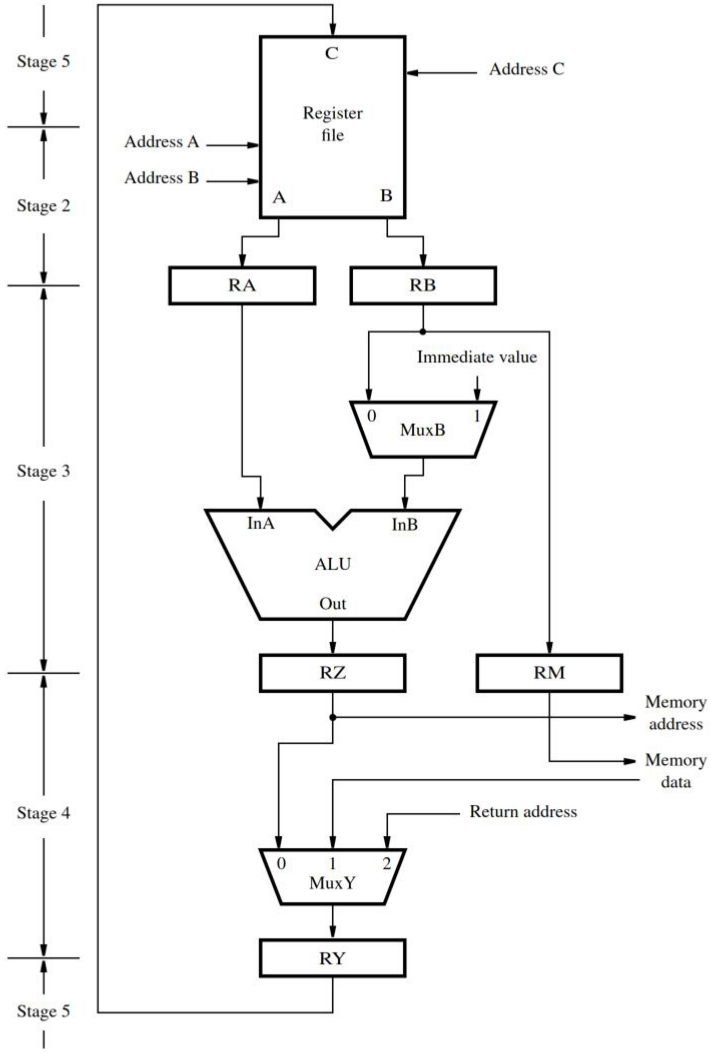
Offset comes from immediate field within the IR, extended by the "Immediate" block

Output routed to PC via MuxPC

RA – subroutine linkage
PC Temp holds return address

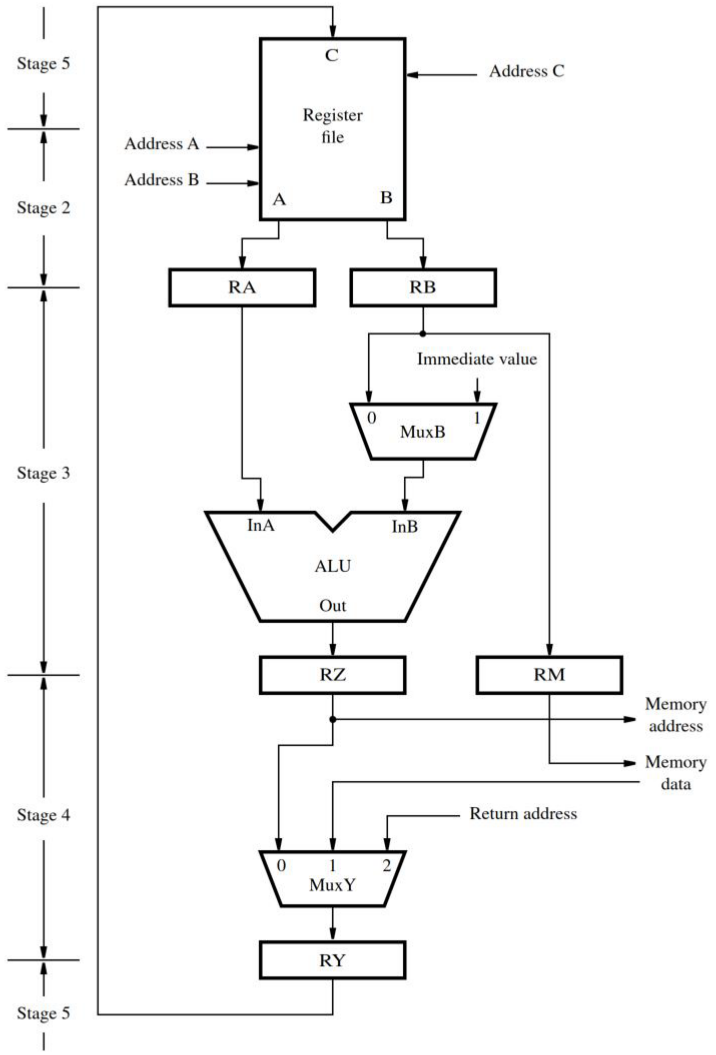
Processing Unit

- Example: ADD R3, R4, R5
  - 1) mem addr ←PC
    - Read memory
    - IR ← Memory data
    - PC ← PC + 4
  - 2) Decode instruction
    - RA ←[R4]
    - RB ←[R5]
  - 3) Operate
    - RZ ←[RA] + [RB]



Processing Unit

- Example: ADD R3, R4, R5
  - 4) RY ← [RZ]
  - 5) R3 ← [RY]
    - Destination register (in register file)



Processing Unit

- In general
  - Reading from memory takes a lot longer than from the register file
    - Keep a cache nearby – close and fast (particularly if data needed is in the cache…)
  - Branch instructions
    - Branch offset has limited # of bits on the “immediate”
      - Limits the memory to be accessed.