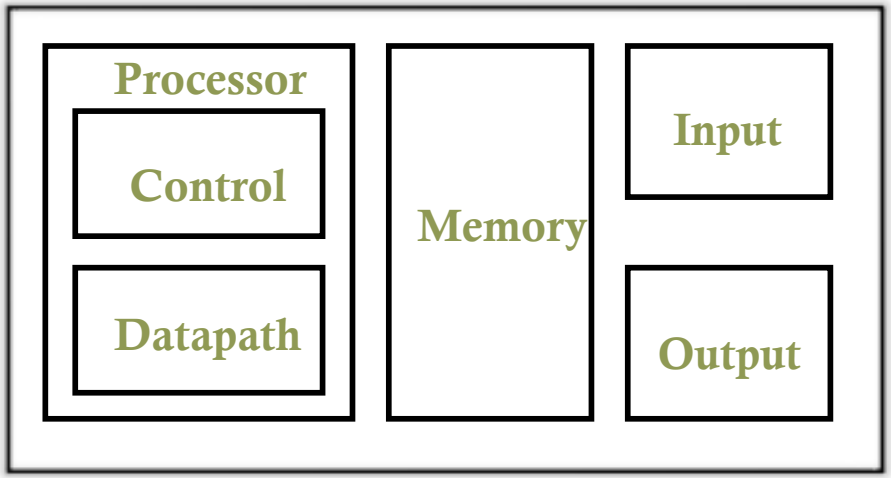
CS465: Computer Systems Architecture

Lecture 6: Processor I – Single Cycle CPU

*Slides adapted from Computer Organization and Design by Patterson and Henessey

Big Picture: Where are We Now?

- Five classic components of a computer



- Today's topic: design a single cycle processor

How to Design a Processor

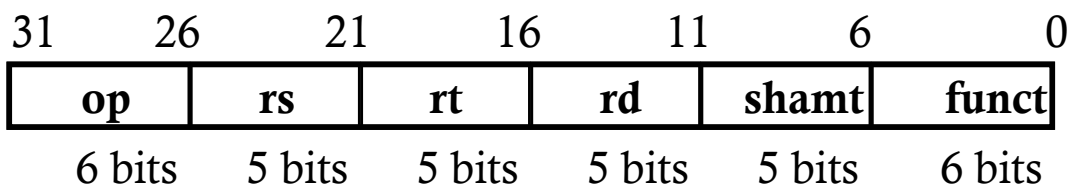
1. Analyze instruction set $\Rightarrow$ datapath requirements
   - Meaning of each instruction given by register transfers
   - Datapath must include storage element for ISA registers (possibly more)
   - Datapath must support each register transfer
2. Select set of datapath components and establish clocking methodology
3. Assemble datapath meeting the requirements
4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer
5. Assemble the control logic

MIPS Lite

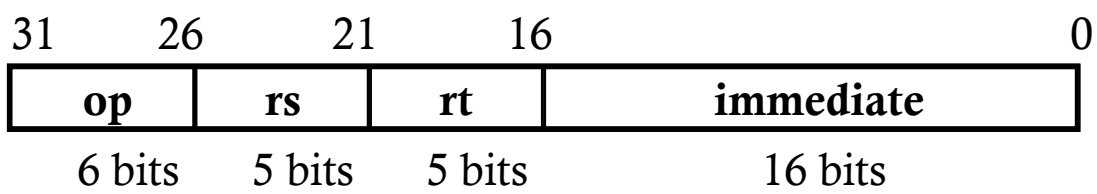
- Simple subset, shows most aspects
  - Memory reference: `lw`, `sw`
  - Arithmetic/logical: `add`, `sub`, `and`, `or`
  - Control transfer: `beq`, `j`

Step 1a: The MIPS-lite Subset

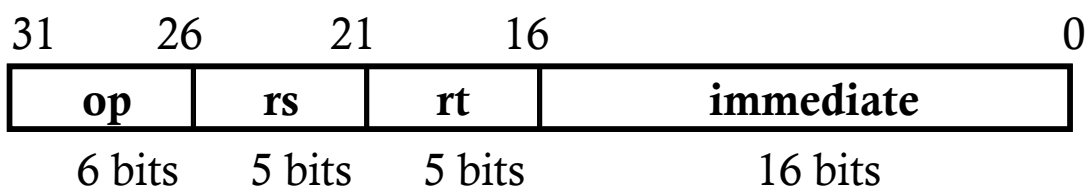
- ADD, SUB, AND, OR
  - add rd, rs, rt
  - sub rd, rs, rt
  - and rd, rs, rt
  - or rd, rs, rt



- LOAD and STORE Word
  - lw rt, rs, imm16
  - sw rt, rs, imm16



- BRANCH:
  - beq rs, rt, imm16



Register Transfer Language

- RTL gives the meaning of the instructions
- First step is to fetch/decode instruction from memory

op | rs | rt | rd | shamt | funct = MEM[PC]

op | rs | rt | Imm16 = MEM[PC]

inst	Register Transfers	
ADD	R[rd] <– R[rs] + R[rt];	PC <– PC + 4
SUB	R[rd] <– R[rs] – R[rt];	PC <– PC + 4
OR	R[rd] <– R[rs] | R[rt];	PC <– PC + 4
LOAD	R[rt] <– MEM[R[rs] + sign_ext(Imm16)];	PC <– PC + 4
STORE	MEM[R[rs] + sign_ext(Imm16)] <– R[rt];	PC <– PC + 4
BEQ	if (R[rs] == R[rt]) then PC <– PC +4 + (sign_ext(Imm16) <<2)	
		else PC <– PC + 4

Step 1b: Requirements of Instr. Set

- PC → instruction memory, fetch instruction
- Register numbers → register file, read registers
- Depending on instruction opcode/funct
  - Use ALU to calculate
    - Arithmetic result
    - Memory address for load/store
    - Value comparison / Branch target address
  - Access data memory for load/store
  - PC ← target address or PC + 4

How to Design a Processor

1. Analyze instruction set ⇒ datapath requirements
   - Meaning of each instruction given by register transfers
   - Datapath must include storage element for ISA registers (possibly more)
   - Datapath must support each register transfer
2. Select set of datapath components and establish clocking methodology ←
3. Assemble datapath meeting the requirements
4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer
5. Assemble the control logic

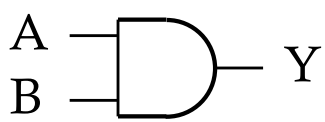
Step 2: Components of Datapath

- Combinational elements
  - Operate on data
  - Output is a function of input
- State(sequential) elements
  - Store information

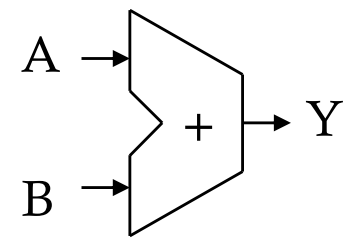
Combinational Elements

Basic building blocks

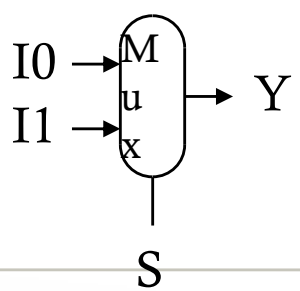
- AND gate
  - Y = A & B



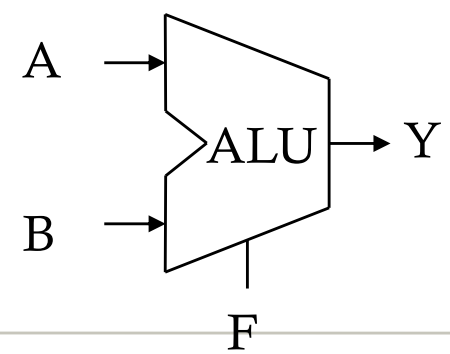
- Adder
  - Y = A + B



- Multiplexer
  - Y = S ? I1 : I0

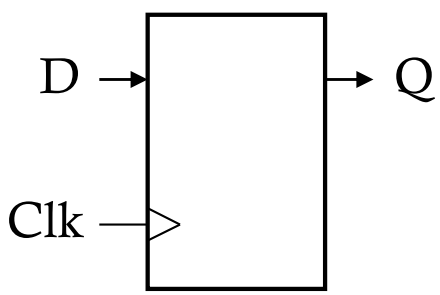


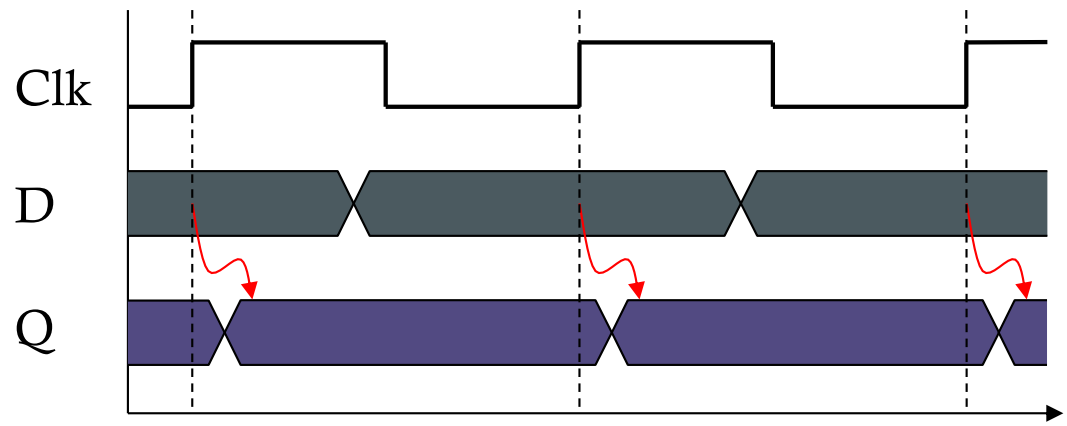
- Arithmetic/Logic Unit
  - Y = F(A, B)



Sequential Elements

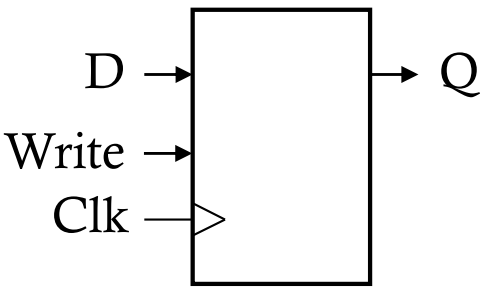
- Register: stores data in a circuit
  - Uses a clock signal to determine when to update the stored value
  - Edge-triggered: update when Clk changes from 0 to 1

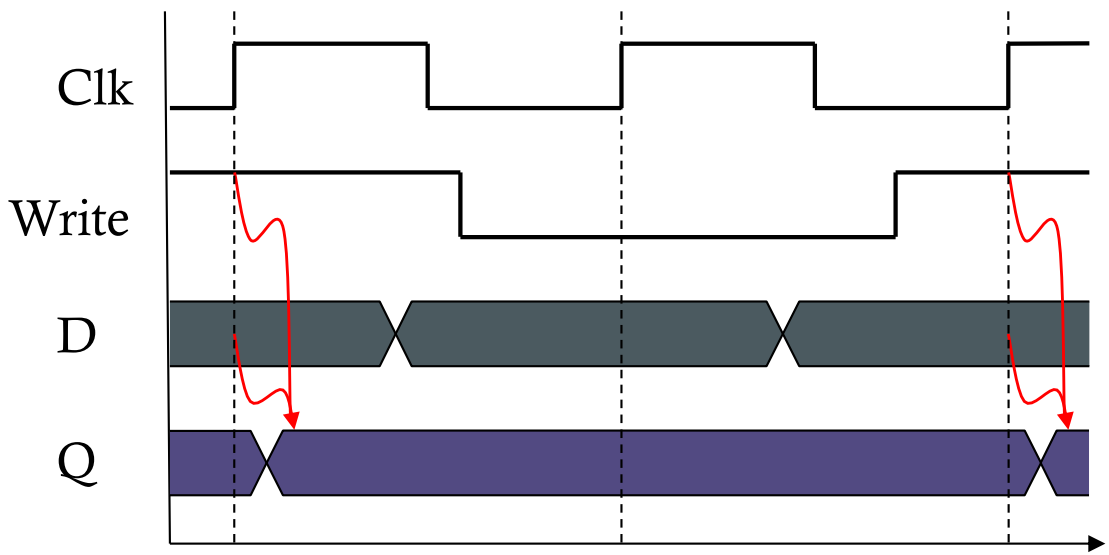




Sequential Elements

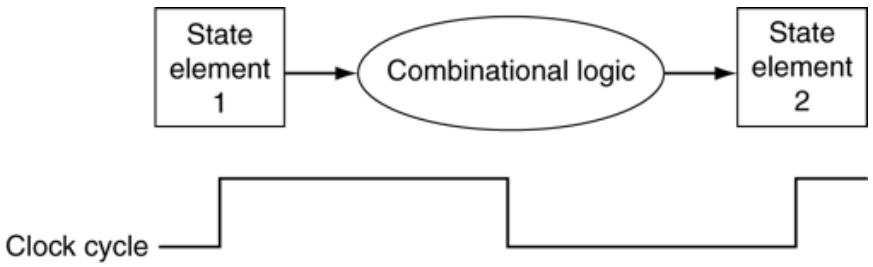
- Register with write control
  - Only updates on clock edge when write control input is 1
  - Used when stored value is required later

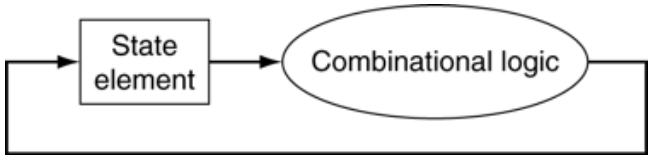




Clocking Methodology

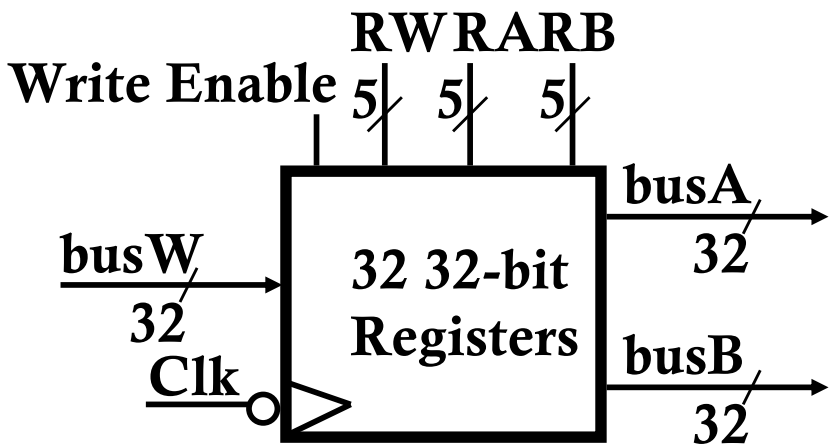
- Combinational logic transforms data during clock cycles
  - Between clock edges
  - Input from state elements, output to state element
  - Longest delay determines clock period





Storage Element – Register File

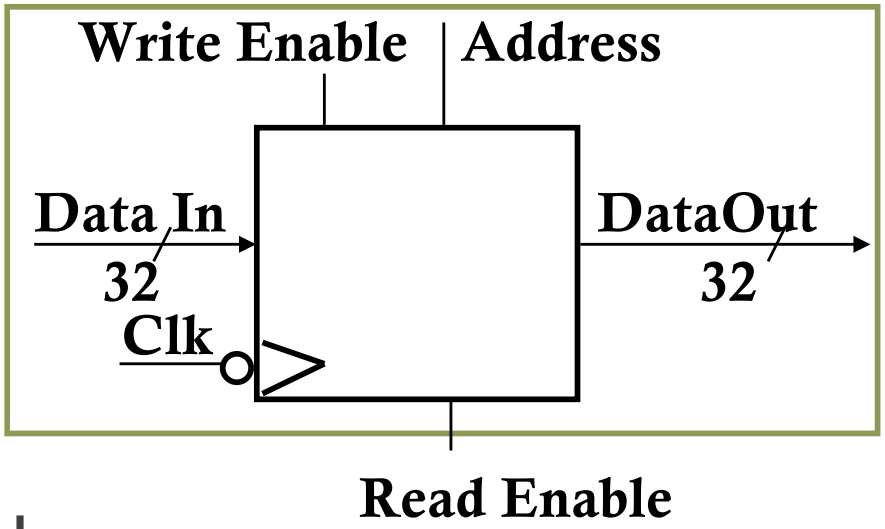
- Register file consists of 32 registers:
  - Two 32-bit output buses:
    - busA and busB
  - One 32-bit input bus: busW



- Register is selected by:
  - RA (number) selects the register to put on busA (data)
  - RB (number) selects the register to put on busB (data)
  - RW (number) selects the register to be written via busW (data) when Write Enable is 1

Storage Element: Memory

- Memory
  - One input bus: Data In
  - One output bus: Data Out



- Memory word is selected by:
  - Address selects the word to put on Data Out
  - Write Enable = 1: address selects the memory word to be written via the Data In bus
  - Similar idea for Read Enable

Roadmap

- 1. Analyze instruction set ⇒ datapath requirements √
  - MIPS Lite: arithmetic/logical, data moving, branch
- 2. Select set of datapath components and establish clocking methodology √
  - Combinational and sequential elements
- 3. Assemble datapath meeting the requirements
- 4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer
- 5. Assemble the control logic

Building a Datapath

- Datapath
  - Elements that process data and addresses in the CPU
    - Registers, ALUs, mux's, memories, …
- We will build a MIPS datapath incrementally
  - Fetch instructions
  - Read operands and execute instructions

Register Transfer Language

- RTL gives the meaning of the instructions
- First step is to fetch/decode instruction from memory

op | rs | rt | rd | shamt | funct = MEM[PC]

op | rs | rt | Imm16 = MEM[PC]

← Instruction Fetch

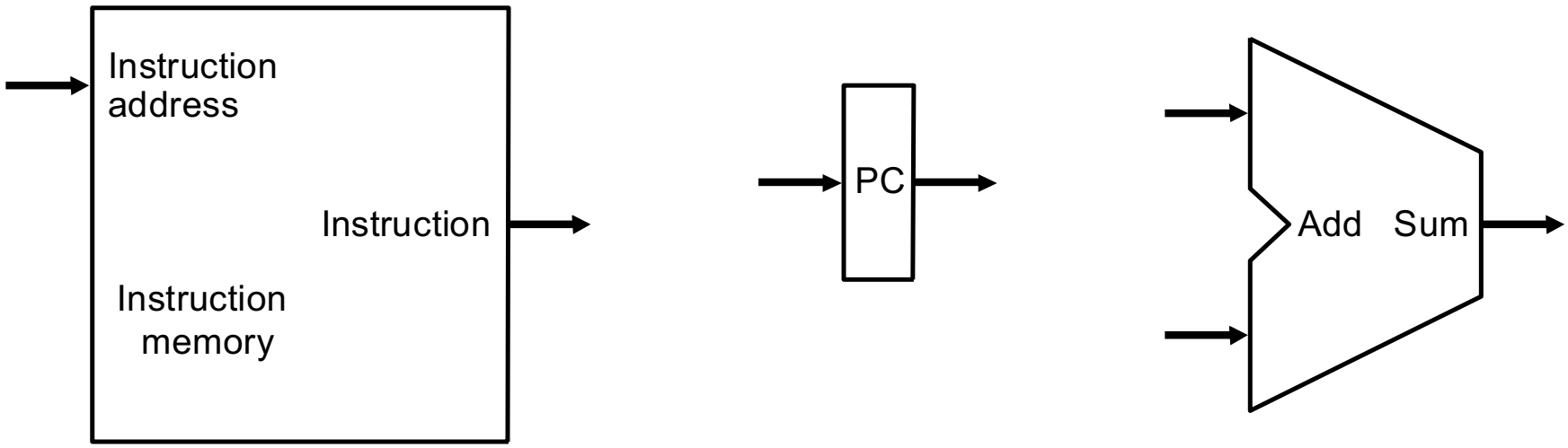
PC updating

inst	Register Transfers	
ADD	R[rd] <– R[rs] + R[rt];	PC <– PC + 4
SUB	R[rd] <– R[rs] – R[rt];	PC <– PC + 4
OR	R[rd] <– R[rs] | R[rt];	PC <– PC + 4
LOAD	R[rt] <– MEM[R[rs] + sign_ext(Imm16)];	PC <– PC + 4
STORE	MEM[R[rs] + sign_ext(Imm16)] <– R[rt];	PC <– PC + 4
BEQ	if (R[rs] == R[rt]) then PC <– PC +4 + (sign_ext(Imm16) <<2)	
		else PC <– PC + 4

3a: Instruction Fetch Unit

- The common RTL operations
  - Fetch the instruction: mem[PC]
  - Update the program counter:
    - Sequential code: PC <- PC + 4
    - Branch and jump: PC <- "something else"
    - We don't know if instruction is a branch/jump or one of the other instructions until we have fetched and interpreted the instruction from memory
    - So all instructions initially increment the PC by 4

Components to Assemble



a. Instruction memory

b. Program counter

c. Adder

Datapath for Instruction Fetch