TED ÜNİVERSİTESİ

CMPE361 Computer Organization

Department of Computer Engineering
TED University. Fall 2023

Microarchitecture

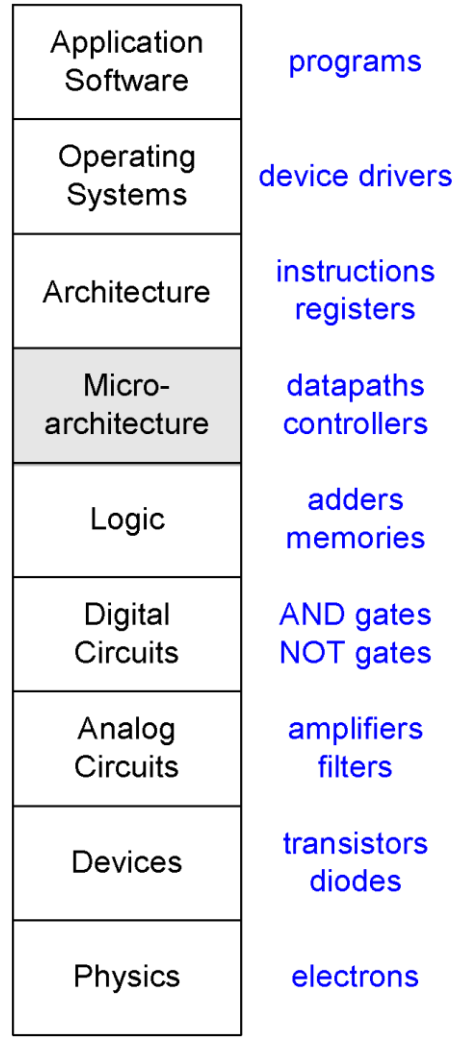
These Slides are mainly based on slides of the text book (downloadable from the book’s website).

How to Implement a Microarchitecture

Microarchitecture is the specific arrangement of registers, ALUs, finite state machines (FSMs), memories, and other building blocks needed.

Implementing Microarchitecture

- Backgroun assumed: familarity with
  - combinational and sequential logic,
  - Memory and arithmetic modules,
  - MIPS architecture: instructions and addressing.



Implementing Microarchitecture

- Multiple implementations for a single architecture are possible
- In this course two implementations are attempted:
  - **Single-cycle:**

    Each instruction executes in a single cycle

  - **Pipelined:**

    Each instruction broken up into series of steps, so that multiple instructions can be executed in parallel

Architectural State

- State of an architecture refers to the content of its components such as PC, Registers, and memory, where:-
  - PC

    address of the instruction to be executed
  - 32 registers:

    Current content at the time of the current instruction execution
  - Memory:

    Current content, including **static data, heap, stack and text** areas at the time of the current instruction execution …

MIPS Processor

- To keep it simpler, we consider only a subset of the MIPS instruction set.
  - R-type instructions: `and`, `or`, `add`, `sub`, `slt`
  - Memory instructions: `lw`, `sw`
  - Branch instructions: `beq`
  - An I-type instructun `addi` and
  - J-type instruction `j`.

Microarchitecture design process

- The processor design is structured into 2 interacting parts:
  - **Datapath:**
    - Operates on data words
    - Includes components such as memories, registers, adders, multiplexers.
    - Maximum length of data path In MIPS is 32 bits.
  - **Control:**
    - control signals tells the datapath what to do with the data.
    - Sample signals subset could be **multiplexer select, register enable, memory write enable** etc.

Microarchitecture design approach

- Start with the current state.
- Use combinational logic to arrive at the next state.
- Let the state elements change their state on the rising edge of the clock.
- So the architecture is based on **synchronous sequential logic**.

Description of state components (hardware blocks)

Main components :

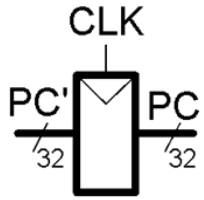
- PC block
- registers block or register file
- instruction memory block
- data memory block

PC block

The PC is an ordinary 32-bit register.

- Assume its
  - Output (PC) indicates the address of the current instruction.
  - Input (PC') indicates the address of the next instruction.
- It has a Clock input to synchronize

MIPS State Elements

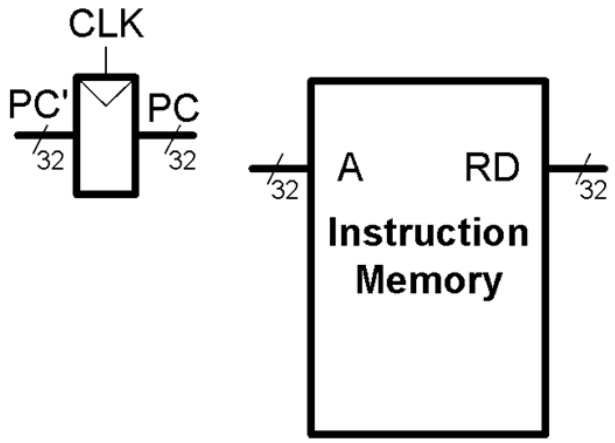


Instruction Memory block

The instruction memory

- takes a 32-bit instruction address (32bits) as input (on port A);
- Transfers the 32-bit instruction to the 32 bit output (port RD).

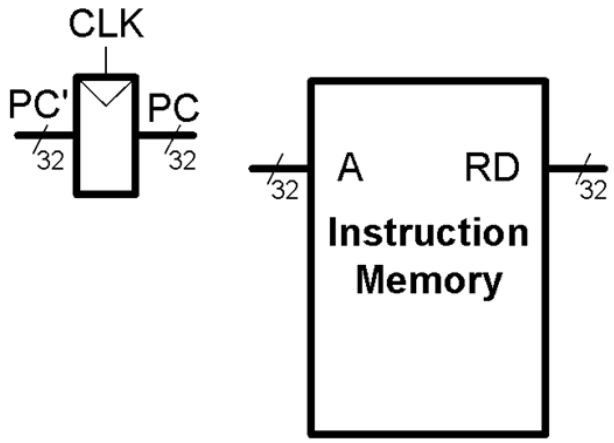
MIPS State Elements

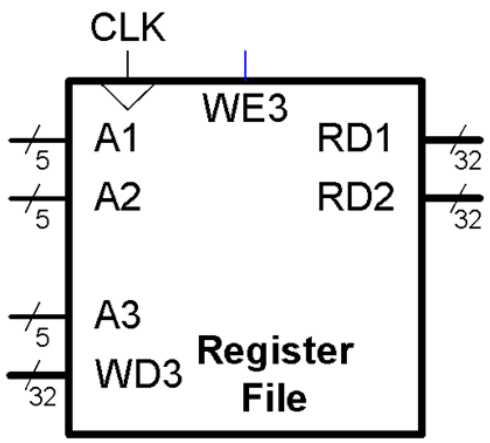


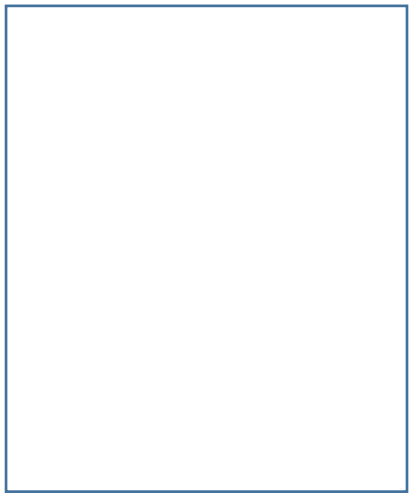
Registers block or Registers file

- The registers block has:-
- Inputs A1 and A2, each 5 bits, specifying one of 32 registers as source operands.
- The value of register indicated on A1 and A2 are copied onto data outputs RD1 and RD2 respectfully.
- The write data port WD takes a 32-bit input and write to register indicated by input port A3.
  - It needs a write enable input, WE3; and a clock CLK.
    - If the write enable WE3 is 1, write the data into the specified register (by A3), on the rising edge of the clock CLK.

MIPS State Elements





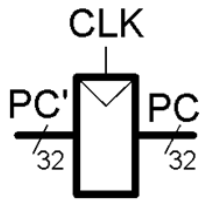


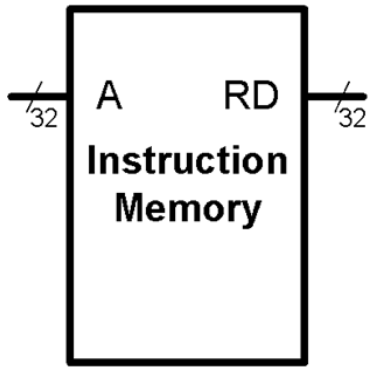
Read is combinational operation
Write is performed on rising edge of the clock.

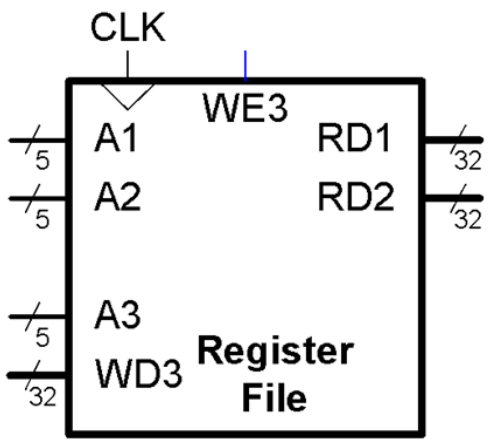
Data Memory block

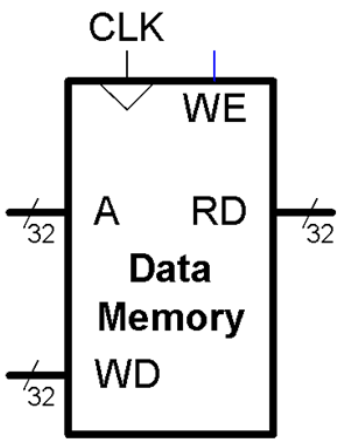
- The data memory has a single read/write port.
- Memory write: If the write enable, WE, is 1, it writes data on 32-bit port WD into address indicated on 32-bit port A, on the rising edge of the clock CLK.
- Memory read: If the write enable is 0, it reads memory address on port A to output port RD.

MIPS State Elements









WE=0, Read
WE=1, Write on rising clock edge