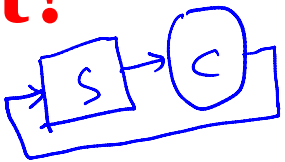
CS305
Computer Architecture

Analysis of the Single Cycle MIPS Implementation

Bhaskaran Raman
Room 406, KR Building
Department of CSE, IIT Bombay

http://www.cse.iitb.ac.in/~br

Single Cycle Implementation: When to Activate Each Element?



- Edge triggered versus level triggered?
  - Different (sequential) elements must trigger at different delays with respect to the start of an instruction fetch+execution
  - This can happen only in a level triggered implementation
- Uncontrolled state change with level triggering?

  - Make PC-write level triggered with latter half of cycle
  - Also make RegWr level triggered with latter half of cycle
  - Also make MemWr level triggered with latter half of cycle

Single Cycle Implementation is Inefficient

- Additional delay in clock cycle
  - Each instruction must be as long as instruction with longest delay!
  - E.g. `j` versus `lw`
- Cost: several additional pieces of hardware
  - I vs D memory separation
  - Adder for PC+4 is separate
  - Adder for BrTgt is separate

Illustrating the Inefficiency

- Main components:
  - Instruction memory
  - Register file
  - ALU
  - Data memory
- Say, each component takes 100 pico-sec
- What is the max clock frequency?

0.35 x 400 + 0.25 x 400 + 0.2 x 500 + 0.2 x 300

Components involved in:
- Register-register instructions
- **`sw`**
- **`lw`**
- **`beq`**
- **`j`**

`lw` is longest: 5 x 100 ps = 0.5 ns
Max clock frequency = 2 GHz

Suppose we (magically) have variable clock cycles?
Instruction mix: R 35%, sw 25%, lw 20%, beq 20%
Avg. time per instrn. = 400 ps

Summary

- Single cycle implementation
  - Higher cost than what looks optimal
  - Slower than what looks optimal
- Multi-cycle implementation addresses both
  - Reuse of hardware components
  - Each instruction takes only as long as needed