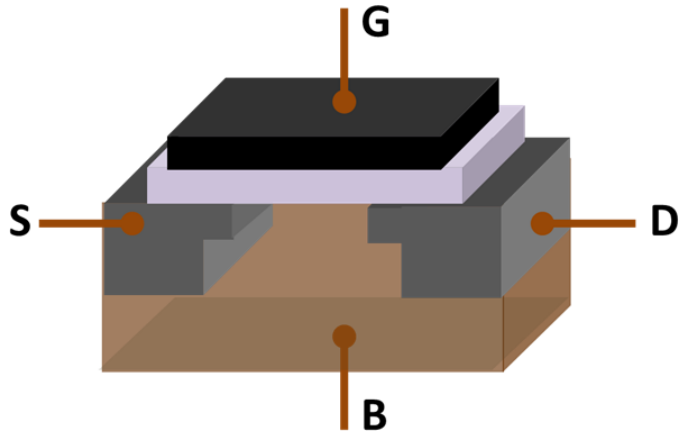
RISC Architecture

Geeth Karunarathne

Lecture Outline

1. RISC Architecture
2. Data Dependency
3. Registers

Speed of Execution



- Increase Frequency
  - More power
  - More heat
  - Hardware limitations
- Improved Architecture
  - RISC
  - CISC

Instruction Set

- The instruction set, also called ISA (instruction set architecture), is part of a computer that pertains to programming in machine language.
- The instruction set provides commands to the processor.
- The instruction set consists of addressing modes, instructions, native data types, registers, memory architecture, interrupt, and exception handling, and external I/O.

Microprocessor Architecture

- RISC (Reduced Instruction Set Computer)
- CISC (Complex Instruction Set Computer)

What is RISC

- RISC – Reduced Instruction Set Computer
- Microprocessor architecture that utilizes a **small, highly optimized set of instructions.**
- More specialized set of instructions found in other types of architectures

Common Features of RISC Architecture

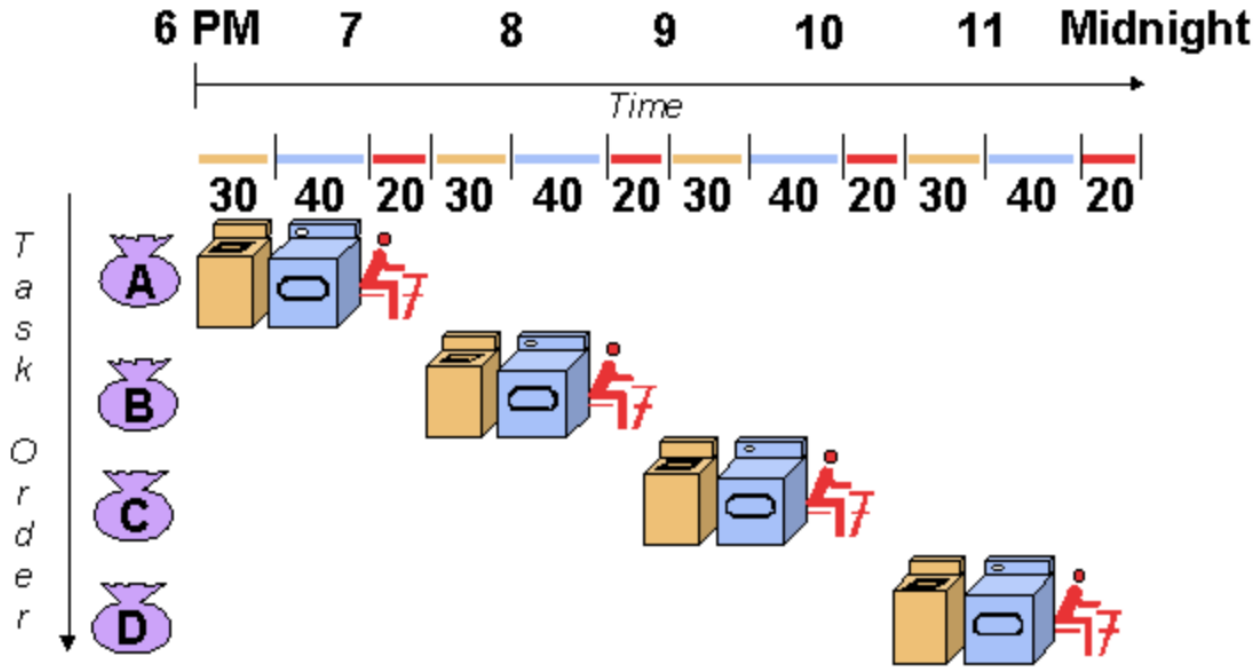
- One instruction per clock cycle
- Pipelining
- Large number of registers

Pipelining

Example…

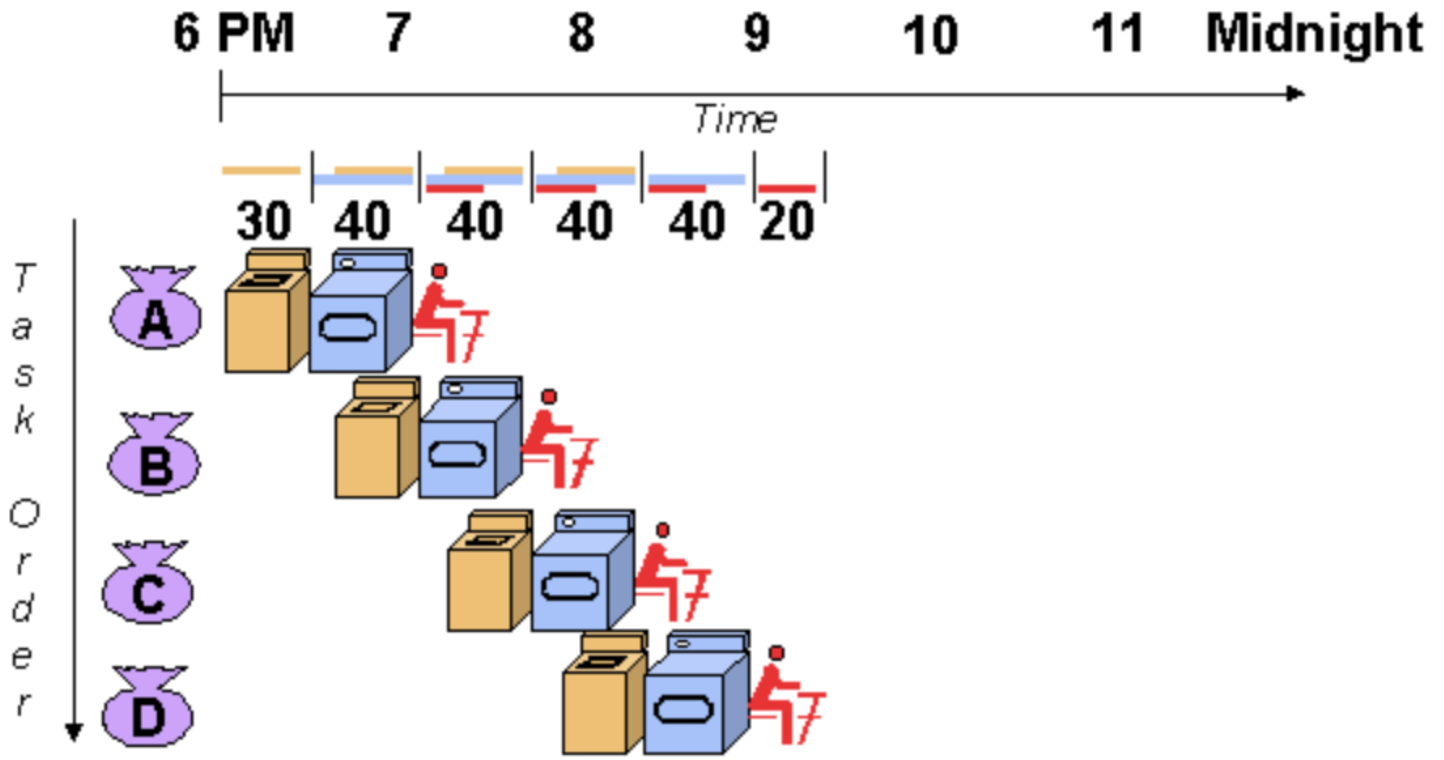
- Laundry Service
  - Wash (30 mins)
  - Dry (40 mins)
  - Fold (20 mins)
- Four Loads of clothes need to washed, dried and fold.

- The conventional way
  - Finish one load before going to the next.
  - Takes 6 hours to finish all four loads.



Source: http://www.ece.arizona.edu/~ece462/Lec03-pipe/

- Let’s try pipelining
  - Takes only 3 hours and 40 minutes.

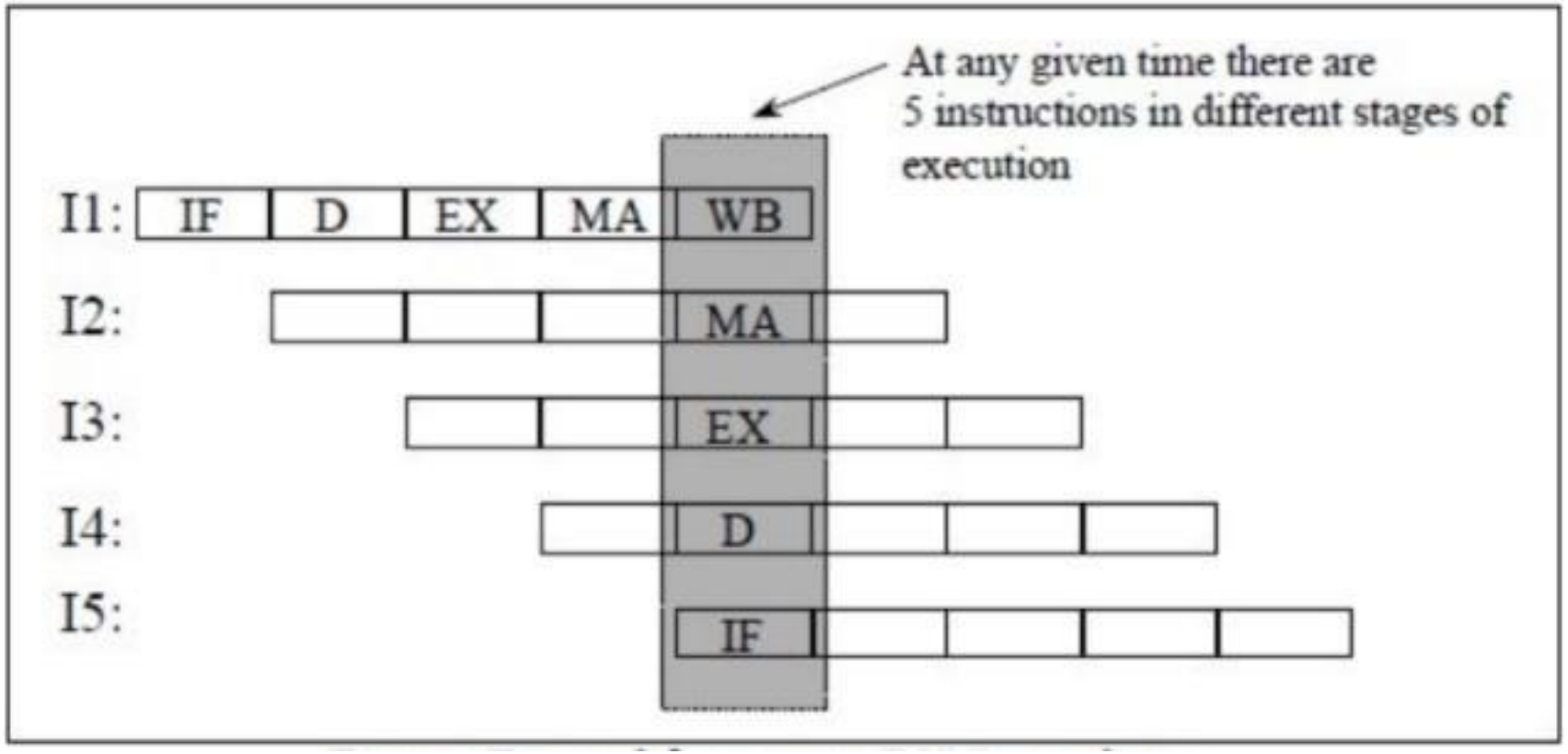


Source http://www.ece.arizona.edu/~ece462/Lec03-pipe/

RISC Pipelines

- 5 basic stages
  - **Fetch** instructions from memory
  - Read Registers and **decode** the instruction
  - **Execute** the instruction
  - **Access** an operand in data **memory**
  - **Write** the **result** into a register
- Ideally, each stage in pipeline should take one clock cycle to execute
- In practice, it may take longer than one cycle per instruction

RISC Pipeline



IF – Instruction Fetch
ID – Instruction Decode
EX – Execute
MA – Memory Access
WB – Write Back

Data dependency (stalling)

```
add $r3, $r2, $r1
add $r5, $r4, $r3
...
```

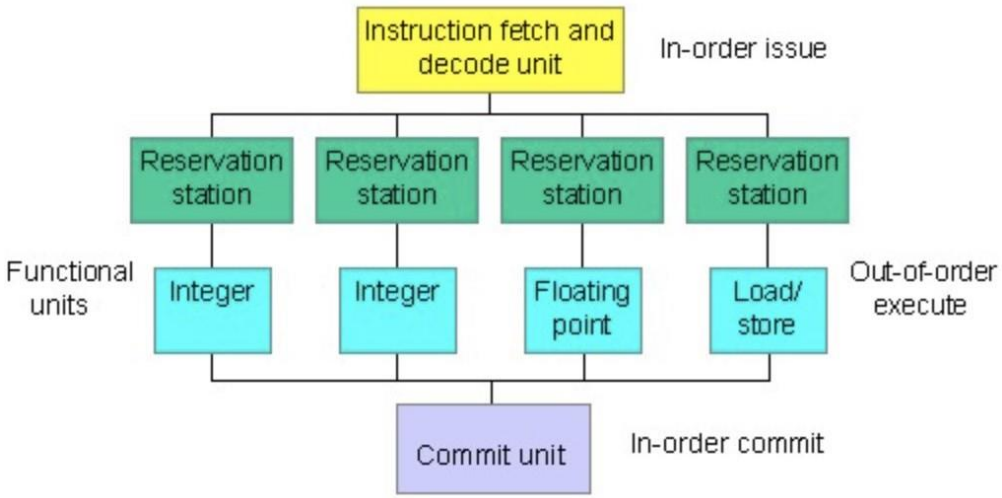
- Reading R3 in second instruction has to wait till first instruction writes it's result to R3
- Data dependency affects long pipelines since it'll take a longer period of time to reach final register writing stage.
- **Code reordering:** If other instructions has nothing to do with the first 2, they are run in between.

Pipeline developments

- Superpipelining
  - Dividing pipeline into more steps
  - More pipe steps, faster the pipeline
  - e.g.
    - Instruction fetch (1st half)
    - Instruction fetch (2nd half)
    - Register fetch
    - Instruction execute
    - Data cache access (1st half)
    - Data cache access (2nd half)
    - Tag check
    - Write back
  - Instruction fetch and data cache access divided into 2 stages

- Superscalar pipelining
  - Multiple stages of all/some of the pipeline stages
    - Internal components of the processor are replicated
    - Handle several instructions in parallel

- Dynamic pipelines: capability to schedule around stalls
  - Instruction fetch and decode unit
  - Functional units
    - Reservation station: buffer holds operands and operations
  - Commit unit



Registers

- Storage space for units of memory
  - Hold temporary values while working on longer strings of instructions
- Transfer data for immediate use
- Common types
  - Memory address register
  - Memory buffer register
  - Input output address register
  - Input output buffer register
  - Shift register

Register Windows

- Improve performance of procedure calls
  - Computer programs compete for the use of registers
  - A function may attempt to use all registers visible
- A set of registers that are visible (allocated) to the function
- Other registers are made invisible

Register Renaming

- Abstracts logical registers from physical registers
- Eliminates false data dependency

Thank You...!