Royal Holloway University of London

Lecture 11 – Other Processor Architectures

Karl R. Wilcox
Karl@cs.rhul.ac.uk

Royal Holloway University of London

Objectives

- **In this lecture we will cover**
  - **A review of the course to date**
  - **Other processor architectures**
    - **Superscalar**
    - **VLIW**
    - **Parallel**

Review - 1

- **We looked at state machines:**
  - **State transition diagrams**
  - **Traffic light control system**
- **State machine implementation**
  - **Using ROMs, PLAs**
  - **Other programmable logic devices**
- **State machine minimisation**
  - **Karnaugh Maps**
  - **Quine McCluskey**

Review – 2

- **We looked at processor instruction sets**
  - **The MIPS instruction set**
  - **Logic for implementing instructions sets**
  - **Built from familiar components**
    - **Registers (banks of flip-flops)**
    - **Arithmetic Logic Units (Adders)**
    - **Control signals (generated by state machines)**

Review - 3

- **We looked at processor architectures**
  - **Single cycle datapath**
    - **1 (long) clock per instruction**
  - **Multi-cycle datapath**
    - **Shorter clocks, as many as needed for each instruction**
  - **Pipelined datapaths**
    - **Overlapping execution of instruction steps**

Review – 4

- **Problems with pipelining**
  - **Hazards**
    - **Structural (resolve by design)**
    - **Control (early evaluation / stall / branch delay slot)**
    - **Data (stall / register forwarding)**
  - **Interrupt Handling**
    - **Logic for interrupt handling**
    - **Instruction selection**

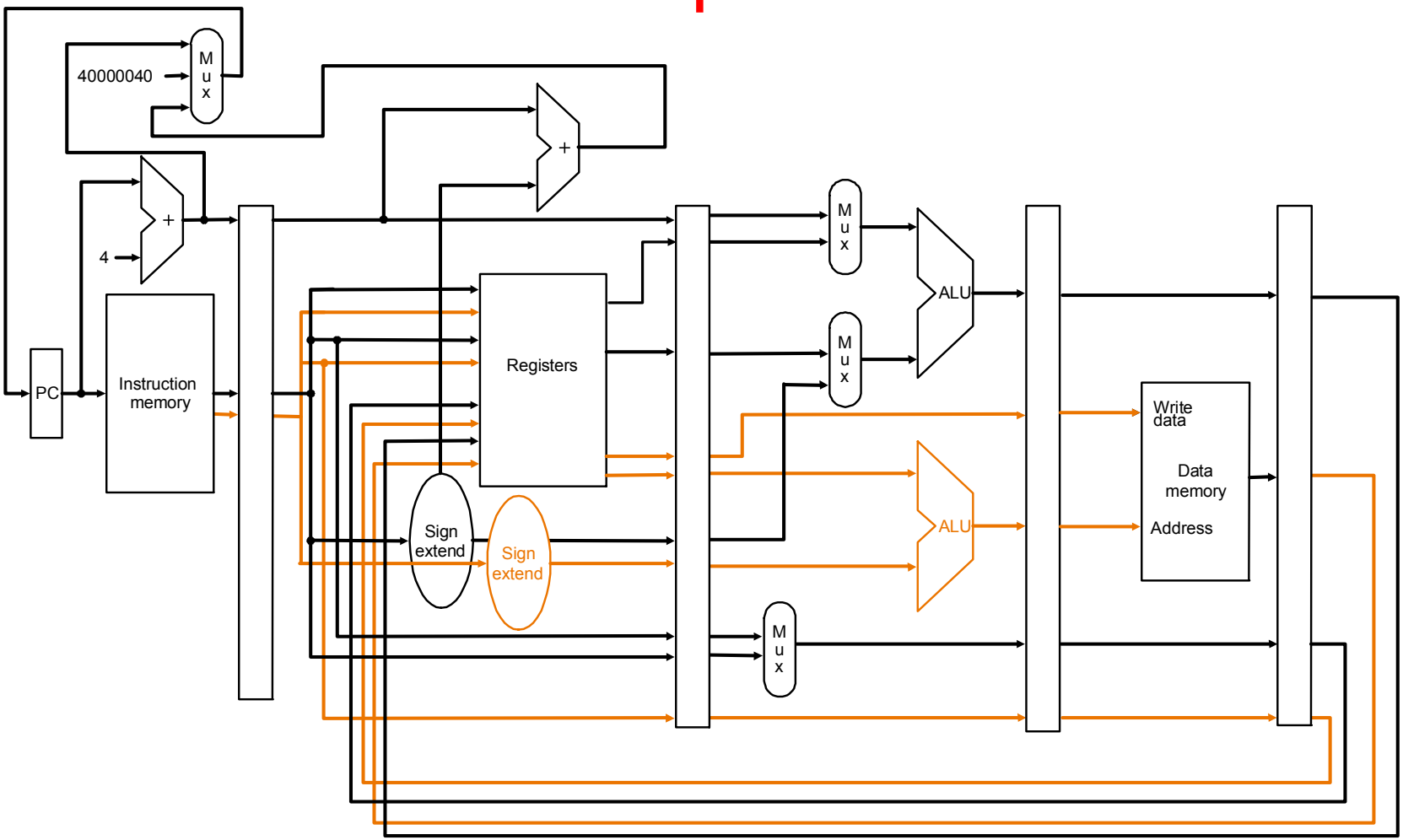
Longer Pipelines

- **Break instructions into more steps**
- **Theoretical maximum improvement in throughput is pipeline length**
- **But:**
  - **Data hazards are more likely (more overlapping instructions)**
  - **Higher penalties for control hazards**
  - **Overhead of extra logic and registers**

Superscalar Processors

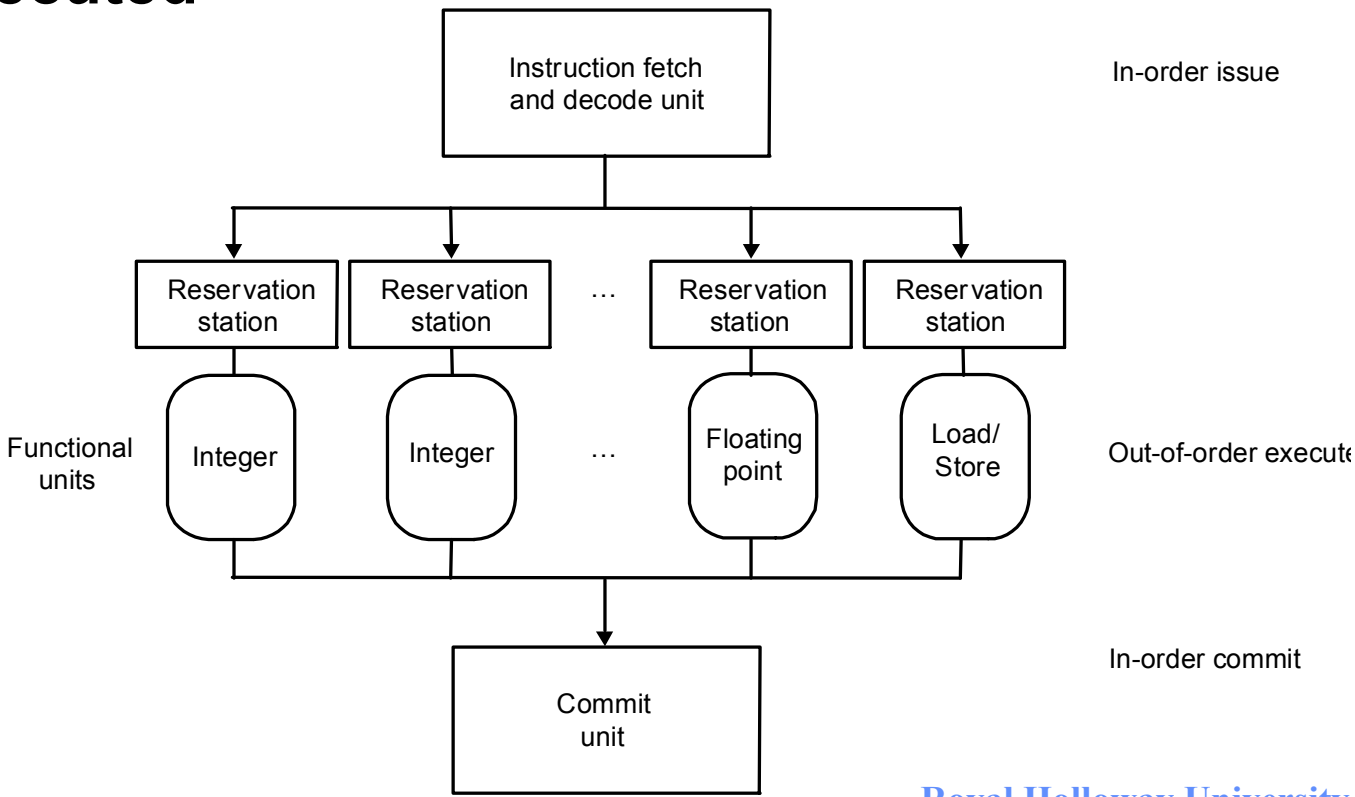
- Replicate the (pipelined) datapath logic
- Launch multiple instructions per clock cycle
- Typically, 2 – 6 instructions per pipeline stage
  - Possibly by type (e.g. 1 floating point + 1 other)
- Can only do this if instructions are independent
- Scheduling by hardware (invisible to compiler)

Two Instruction Superscalar MIPS



Dynamic Pipeline Scheduling

- **Instead of stalling, find an instruction that can be executed**



Reservation Station & Commit Unit

- **The reservation station:**
  - **Holds the operation and the operands**
  - **Executes the operation when all are available**
- **The commit unit:**
  - **Writes registers / memory when all previous instructions have been committed**

Further Complications

- **Dynamic pipelining with branch prediction is called *speculative execution***
- **Usually combined with superscalar so more than one instruction can be committed per clock cycle**
- **Makes processor design very complicated**
  - **and difficult to prove correct**

Very Long Instruction Word

- **A fixed number of parallel operations is combined into a single instruction word**
  - **E.g., 2 integer operations, 2 FP ops, 2 Memory refs, 1 branch**
- **The compiler must schedule instructions (including delays if required) to get the right result**
  - **The pipeline is exposed to the compiler**
- **Aka Explicit Parallel Instruction Computer**

Vector Processor

- **Single Instruction – Multiple Data (SIMD)**
- **One instruction executed simultaneously on more than one item of data**
- **Common in multi-media hardware**
  - **Graphics cards**
  - **Pentium MMX extensions**

Multiple (Parallel) Processors

- **Multiple Instruction Multiple Data (MIMD)**
- **Complete Duplication of Processor**
- **Could share a single memory space**
- **Or communicate over a bus**
  - **In either case need to *synchronise***
- **Not all problems are amenable to parallel solutions**

Royal Holloway University of London

Summary

- We can (hopefully!) see how a complex processor is built out of simple digital logic
- We have looked in detail at pipelining processors
- We looked briefly at other processor architectures
  - Superscalar
  - Dynamic Pipeline
  - Vector
  - Parallel

Next Lecture

- **Cache memory systems**
  - **Implementing a memory hierarchy**
  - **(Related to CS263, Operating Systems)**
    - **Not concerned with page replacement algorithms**
    - **Interested with actual hardware implementation of caches**