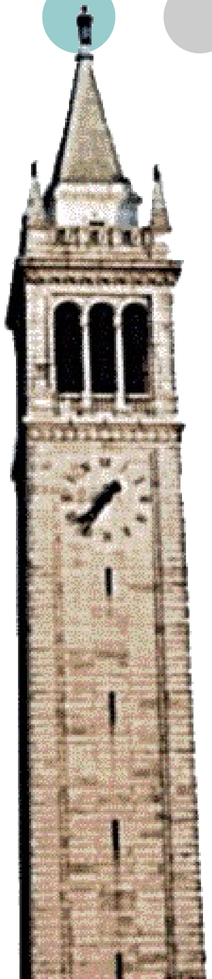
Introduction to Embedded Systems

Week 11: Embedded Processors



Processor architectures

- Instruction set architecture (ISA)
  - = Definition of instructions supported by a processor, + their constraints
  - Example: Intel x86 architecture
- Processor chip
  - = The physical processor created by a manufacturer
- Same ISA = interchangeable processors
  - Multiple manufacturers can create different products with same ISA => they are interchangeable
  - Same code can run on all processors similarly

Types of Processors

- General-purpose processors
  - = processors for general-purpose PCs
- Intel x86 architecture is dominant
  - Intel 8086: 16-bit processor, used in IBM PC => dominance
  - Intel 80386: 32-bit processor
  - x86-64: 64-bit family
  - All backward compatible
- Producers: Intel, AMD, Cyrix …
  - Different products, same architecture

Types of Processors

- Why a common architecture?
  - General-purpose = many different applications need to be used interchangeably, over long time
  - => Need for common instruction,
  - => Need for backward compatibility

Types of Processors

- Embedded processors
  - = processors for embedded systems

- No single dominance, various architectures
  - Embedded = dedicated to a (smaller) single task
  - No need for backward compatibility, changing vendors, etc.
  - => Different architectures dedicated to specific tasks
    - i.e. faster / low-power etc.

Types of Processors

- Microcontrollers
  - = CPU + peripherals on a single chip
  - Peripherals: memory, timers, I/O devices etc.

- Notable families
  - Intel 8051
  - Atmel AVR
  - Microchip PIC

Types of Processors

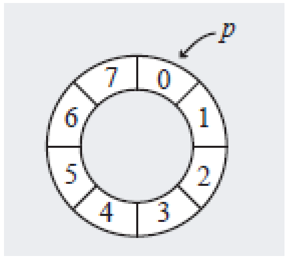
- Digital Signal Processors (DSP)
  - Dedicated for discrete signal processing
- Mathematically intensive operations
  - Digital FIR filtering (convolutions with a impulse response)
  - Fourier transform
- Specialized instructions sets for mathematical operations

Types of Processors

- DSP hardware specifics
  - Efficient **Multiply-and-Add (MAC)** instruction, for FIR filtering
    - Compute an output sample = N MAC operations

$$y[n] = \sum_{i=0}^{N} x[n-i] \cdot h[i] = x[n] \cdot h[0] + x[n-1] \cdot h[1] + \cdots$$

  - **Circular buffers**: for storing last N values efficiently



Types of Processors

- Graphic Processor Units (GPU)
  - Dedicated for intensive graphic computations (triangulation, texture alignment, image/video processing)
  - Extremely efficient for parallel computations
  - Power-intensive, little used in embedded systems

Embedded Processor features

- Parallelism
  - = Possibility of executing different instructions at the same time => faster execution
- Pipelining
  - An instruction is executed in **multiple stages**: A, B, C..
  - While an instruction is stage C, the next one can be already in stage B, and a third one in stage A
    - Faster execution

Example pipelining architecture

- Pipelining architecture with 5 stages
  - (Separated by the grey blocks)

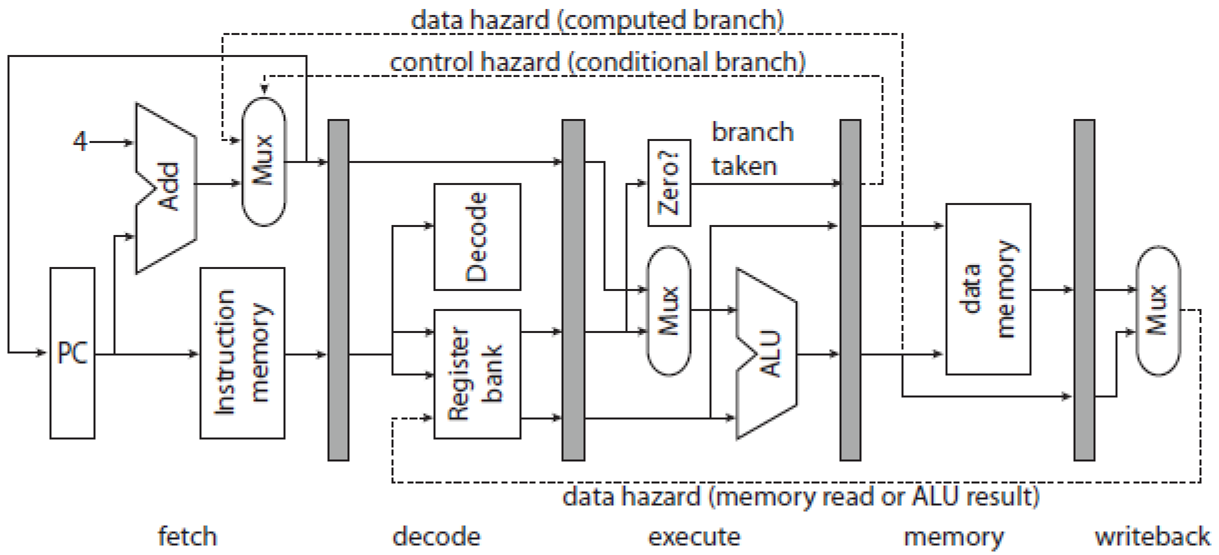


Figure 8.2: Simple pipeline (after Patterson and Hennessy (1996)).

Pipelining problems

- Data hazard
  - Instruction I2 follows after I1 and needs its result
  - Instruction I1 writes output in stage 5
  - But next instruction I2 wants to read it in stage 2, when the previous result is not yet available

- Solution
  - detect this in hardware and artificially delay I2

Pipelining problems

- Control hazard
  - Instruction I1 determines a jump in code (i.e. it is if/else, or function call)
  - Jump is decided when I2 reaches stage 4
  - By that time, the next instructions I2 an I3 are executing in stages 3 and 2, possibly producing effects
  - But I2 and I3 should not have been executed at all
- Solution
  - Speculative execution: detect early a jump instruction
  - Bugs: …

Instruction-level parallelism

- CISC instructions
  - = complex and specialized instructions for certain tasks (e.g. for DSP)

- CISC vs RISC processors:
  - CISC = Complex Instruction Set Computers
    - Many instructions, complex, for dedicated operations
    - Needs complicated compiler to be able to use them under-the-hood
  - RISC = Reduced Instruction Set Computers
    - Few instructions, but very fast
    - Easy to be optimized by a compiler

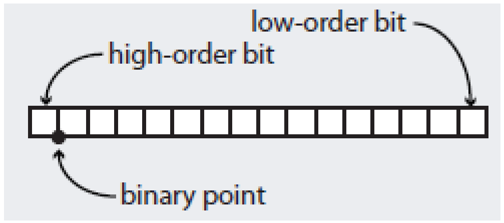
Instruction-level parallelism

- Sub-word parallelism
  - = Pack multiple smaller values in a large register and operate on all of them simultaneously
  - Example: use a 32-bit register for 4 parallel operations on 4 values of 8-bit size (e.g. pixel values)
    - Since all pixels must undergo same processing
  - Especially useful in audio/video processing, where sample values are 8-bit or 16-but long
  - Known as “MMX” instructions on Intel processors

Fixed-point vs floating-point support

- Integer number representation
  - Base 2
  - Negative numbers: in C2 (2’s complement), base 2
  - Example: at blackboard

- Fixed-point fractionary numbers
  - Assume there is a “binary point/comma” at some location
    - Binary point does not exist physically, it is just for computing the value



Fixed-point representation

- If there are F bits after the binary point, the number value is **divided by 2^F**
  - Example at blackboard
    - Write number 15 and number 15/128, on 8 bits
    - Add numbers 15/128 and 20/128 and find the result
    - Convert number 23 from type *int16* to *fixed-point with F=8*
- Advantage of fixed-point
  - Arithmetic = just like integers. Any CPU can use them.
- Disadvantage
  - Range of values is fixed, cannot be adapted on-the-fly
  - Either have lots of decimals, or large numbers, not both

Floating-point representation

- Defined by IEEE 754 standard
- Concept
  - The position of the binary point is mobile (it is encoded in a group of bits, so it can be changed as wished)
    - If position is towards the LSB => large values, fewer decimals
    - If position is towards the MSB => small values, lots of decimals
    - Can accommodate large numbers or very precise decimals
    - Example: C types *float*, *double*
- Disadvantage
  - Needs specialized hardware => larger cost
  - Slower then integer calculations