Politechnika Wrocławska

Computer Architecture and Organization

Lecture 6

Dr. Radosław Michalski
Department of Computational Intelligence, Faculty of Computer Science and Management, Wrocław University of Science and Technology

Version 1.0, spring 2017

Source and licensing

The most current version of this lecture is here:
https://github.com/rmhere/lecture-comp-arch-org

Overview of this lecture

Embedding assembly in C

Demo

Jain.PK - Using Inline Assembly in C/C++

Floating point representation

Integers

How to work with integer numbers in computer systems?

- exemplary integer: 1283093714 (31 bits)
- integers - precise representation
- maximum length - as defined by architecture
- 2^n, where n represents the number of bits
- signed/unsigned
- overflow

Floating point representation

Real numbers - introduction

How to work with real numbers in computer systems?

- exemplary real number: 3.82379102
- no possibility of holding some real numbers precisely
- registers have fixed length (32 bits in case of MIPS)
- precision or approximation
- how to use these 32 bits effectively?
- fixed point vs. floating point

Floating point representation

Real numbers - fixed point

integerpart . *fraction*

3.82379102

00000011 (8 bits) . 1001110100100000000101011110 (28 bits)

Floating point representation

Real numbers - floating point

$$\mathit{significand} * \mathit{base}^{\mathit{exponent}}$$

$$3.82379102 = 382379102 * 10^{-8}$$

Floating point representation

Real numbers - IEEE 754 standard

IEEE 754 / binary32

- sign bit (1 bit)
- exponent (8 bits)
- significand/mantissa (24 bits, 1 bit implicit)
- base: 2

Floating point representation

Real numbers - binary32

$$significand * 2^{exponent}$$

3.82379102

0 (sign)
10000000 (exponent - 1)
11101001011100011111110 (mantissa - 1.9114999771118164)

Floating point representation

Real numbers - IEEE 754 standard

IEEE 754 / binary64

- sign bit (1 bit)
- exponent (11 bits)
- significand/mantissa (53 bits, 1 bit implicit)
- base: 2

Floating point representation

Real numbers - MIPS

- MIPS has 32 single precision (32-bit) floating point registers.
- $f0 – $f31
- $f0 is not special
- special instructions that work on single precision
- these cannot use general purpose registers, only floating point

Floating point representation

Double precision in MIPS

- using the same sets of registers pairwise, e.g., \$f0 and \$f1
- addressing the first register from a pair, e.g., \$f0, \$f2
- instructions for integer, single and double precision arithmetic
  - `add` - integers
  - `add.s` - single precision
  - `add.d` - double precision

Floating point representation

Sources & recommended materials

- S. Hollasch, IEEE Standard 754 Floating Point Numbers (website)
- Wikipedia, IEEE floating point (website)
- H. Schmidt, IEEE-754 Floating Point Converter (website)
- J. King, IEEE Floating Point Standard (The Implicit 1) (video)

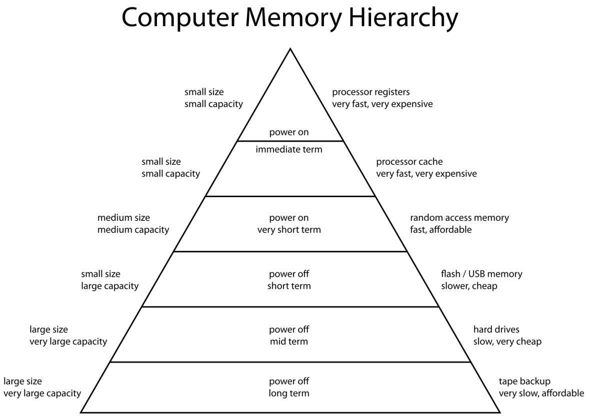
Accessing memory

Introduction

- from computer's perspective (RISC):
  - registers
  - memory
- memory is continuous
- the CPU does not know with what type of memory it interacts
- this is why we can introduce different strategies regarding memory

Accessing memory

Memory hierarchy



Computer memory hierarchy, public domain

Accessing memory

Memory access times

Processor registers:

- 32 * 32 bits (registers) + 32 * 32 bits (floating point registers)
- the fastest, matched in speed to the CPU
- 0.25 ns

Cache:

- megabytes
- 1ns

RAM:

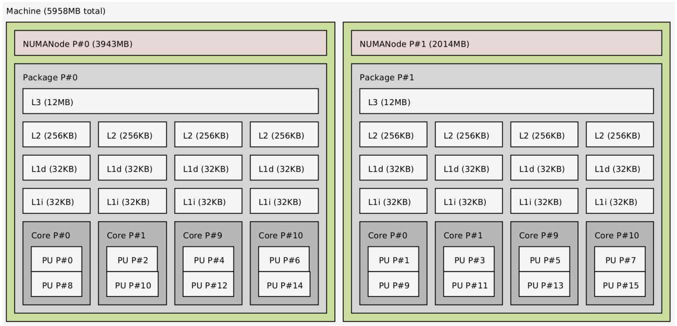
- gigabytes
- 20ns

External memory

Accessing memory

Lstopo output

Lenovo^{TM} ThinkStation® D20, Intel® Xeon^{TM} E5640, 6 GB RAM



Screenshot from the application lstopo (package Portable Hardware Locality)

Accessing memory

Exploring memory hierarchy

From CPU perspective:

- as large as the largest level of hierarchy
- accessed as if it was built from the fastest memory

Levels:

- copying data between levels (lower to upper)
- minimum unit is called a block or a line

Accessing memory

Hits and misses / locality

Finding data/instructions in memory:

- **hit** - block found in a given level
- **miss** - block not found in a given level

Basic management / optimization strategies:

- spatiality
- temporality

Accessing memory

Cache mapping

How to find data in cache?

- **direct mapping** - only one location of block in cache address, tags to identify given block
- **fully associative cache** - any location in cache
- **set-associative** - fixed locations in cache

Accessing memory

Hits and misses ctnd.

- **hit rate** - fraction of memory accesses found in the upper level
- **miss rate** - fraction of memory accesses not found in the upper level
- **hit time** - time to access the upper level of memory and to determine whether it is hit or a miss
- **miss penalty** - time to replace a block in the upper level and time to deliver the block to the processor, stall

Accessing memory

Writing

- **write-through** - write updates both: cache and memory
- **write buffer** - queue holding data to be written to memory
- **write-back** - write data to cache, write to memory only while replacing

Accessing memory

Virtual memory

- efficient and safe sharing of memory among multiple programs
- many programs running at once on a computer and the memory needes is larger than available
- virtual memory - using main memory as a cache for secondary storage

Accessing memory

Sources & recommended materials

- D. Patterson, J. Hennessy, *"Computer Organization and Design"*, Elsevier
- Imagination Technologies Limited, MIPS Software Training - caches, Hertfordshire, UK (training materials)
- J. Kwiatkowski, *"Computer Architecture and Organization"*, Wrocław University of Science and Technology (course materials)

Future of computing and computer architectures

Introduction

Limitations:

- (physics) density of transistors in a silicon wafer
- (physics) copper connectors
- (use) general purpose computing vs. specialization

Directions:

- “*new*” materials
- “*new*” architectures (e.g., Mill)
- “*new*” computing (e.g., quantum, pervasive/ubiquitous, approximate, cognitive)

Future of computing and computer architectures

Videos

HPE Technology: The Machine Photonics and Universal Memory

HACKADAY: Mill CPU for Humans - Preview

Kurzgesagt – In a Nutshell - Quantum Computers Explained – Limits of Human Technology

Big Think - Michio Kaku: Tweaking Moore's Law and the Computers of the Post-Silicon Era

Future of computing and computer architectures

Sources & recommended materials

- G.E. Moore, *“Cramming more components onto integrated circuits”* (scientific article)
- J. Gallego, *“A New Kind of Computer Chip: Silicon May Be Replaced by New Material”* (article)
- K. Boursac, *“Graphene Could Buttress Next-Gen Computer Chip Wiring”*, IEEE Spectrum (article)
- M. Weiser, *“Ubiquitous Computing”* (webpage)
- *Plan 9 from Bell Labs* (webpage)
- *’Approximate computing’ improves efficiency, saves energy*, Purdue University, IN, United States (news)