cs251

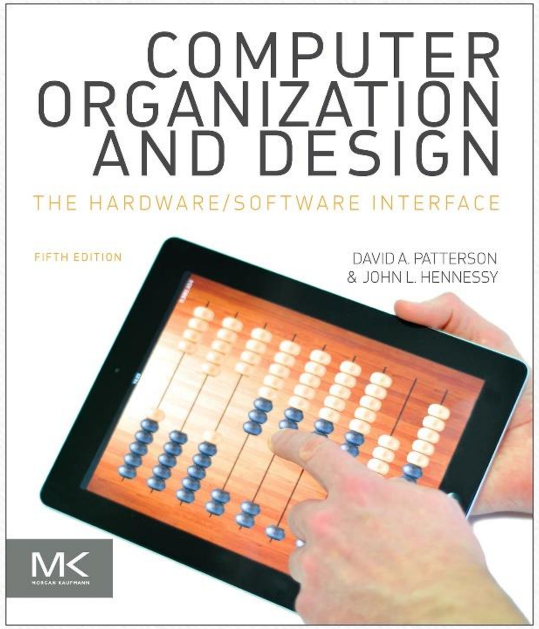
Computer Organization and Design

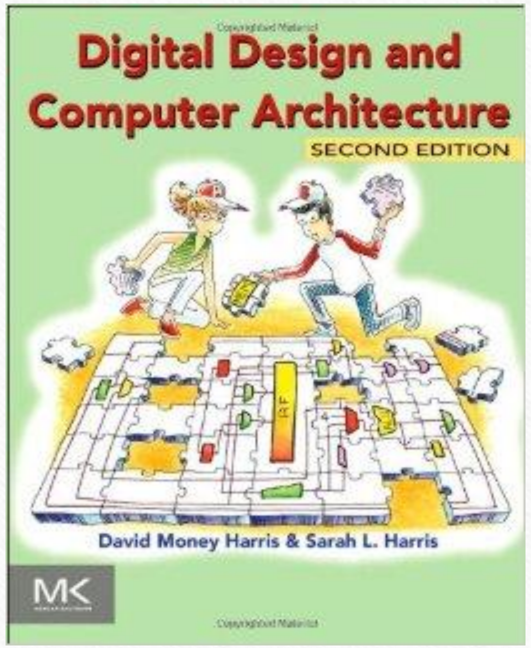
S15 Rosina Kharal

Course Outline : Webpage

*Consult Hours
*Assignments
*Drop Boxes

Texts
Also on Reserve
 DC Library


COMPUTER ORGANIZATION AND DESIGN
THE HARDWARE/SOFTWARE INTERFACE
FIFTH EDITION
DAVID A. PATTERSON
& JOHN L. HENNESSY
MK


Digital Design and Computer Architecture
SECOND EDITION
David Money Harris & Sarah L. Harris
MK

Clickers

- Available at the Bookstore
- Class participation marks 5% of grade
- Top 75% of all your clicker inputs are taken

  * therefore, there is room for co-op interviews

  * illness etc.

Do Not Share clickers

Or Lend to a Friend (click for you during class)

Office Hours

- Rosina Kharal: Beginning Next Week
- Wed: 2:30-3:30 Thursday: TBA DC 2132
  - Or By Appointment
- ISA / IA: Office hours, extra hours during assignment week
- Gaurav Baruah(assignment marking) and Sean Kauffman(monitoring piazza)
- Piazza : Posts answered usually in 24hours

Course Guidelines

- Course Notes: Overview
    - Not comprehensive Not a substitute for lectures
- Textbooks : Additional Study and Reference

- Drop box outside MC 4065 (check labels on box)
- DO NOT SUBMIT ON 3rd FLOOR
- Returned in class

- If you don't want returned in class:
- Email Olga Zorin (olga.zorin@uwaterloo.ca)
- Returned in IA office hours if you don't pick up in class
- Solutions in display case outside MC 4065
- Book authors request that solutions not be online

Excessive Collaboration

In the past, as many as 10%-20% of students in the course have
Been caught and penalized for excessive collaboration.

Previous terms: encouraged to talk/discuss, but must write up solution
on your own without checking with other students.
Excessive similarities treated as excessive collaboration.

This term (S15) Also
Hand in own copy of assignment.

Caution:
Doing assignments is critical for learning the material:
“I feel lost sometimes in lectures, but the assignments help a lot.”

How it All fits together...

- If we tried to build a Usable computer system from scratch

It would certainly overwhelm us (operating system, microprocessor, memory, networking, wireless connectivity etc. etc.)

- Manage the layers of complexity: working at individual layers of the

Computer Architecture

- Understand each layer, and abstract away details of other layers that are not

Important to current layer

- **This Course:**
- Understanding of computer architecture, structure and evolution
- Computer architecture : instruction set architecture plus computer organization
- Instruction set architecture:
  - Conceptual structure and functional behaviour of computing system as seen by programmer
- Computer organization:
  - Physical implementation, described in terms of functional units, their interconnection, how information flow among them is controlled

Course outline

*MIPS review, brief discussion of performance
*Digital logic design
*Data representation and manipulation
*Designing a datapath
*Single-cycle control unit
*Multiple-cycle control units (hardwired and microprogrammed)
*Pipelining and hazards
*Memory Hierarchies (caches and virtual memory)
*Input/Output
*Multiprocessor systems
*Case Studies: VAX, SPARC, Pentium

About You 💻✆

- A) I am CS major
- B) Math major
- C) Engineering
- D) Business or Science
- E) Other...

Computer Organization and Design

- **I want to take this course because:**
- A) I want to learn more about Hardware
- B) I want to learn about inner workings of computer systems
- C) I love CS
- D) I have to take this course- no choice
- E) I want to become a super awesome programmer- and save the world!

Other Courses I have This term

- A) CS241
- B) CS246
- C) Stats
- D) Calculus /Algebra
- E) E&CE Or OTHER

Guiding Principles : Computer Architecture Design

- **Use Abstraction to Simplify Design**
- Moores Law: Expect Rapid Change in Technology
  - IC resources doubles every 18-24 months
  - The number of transistors that can fit onto a circuit board will double. Design your architecture with this in mind...

Make the Common Case Fast

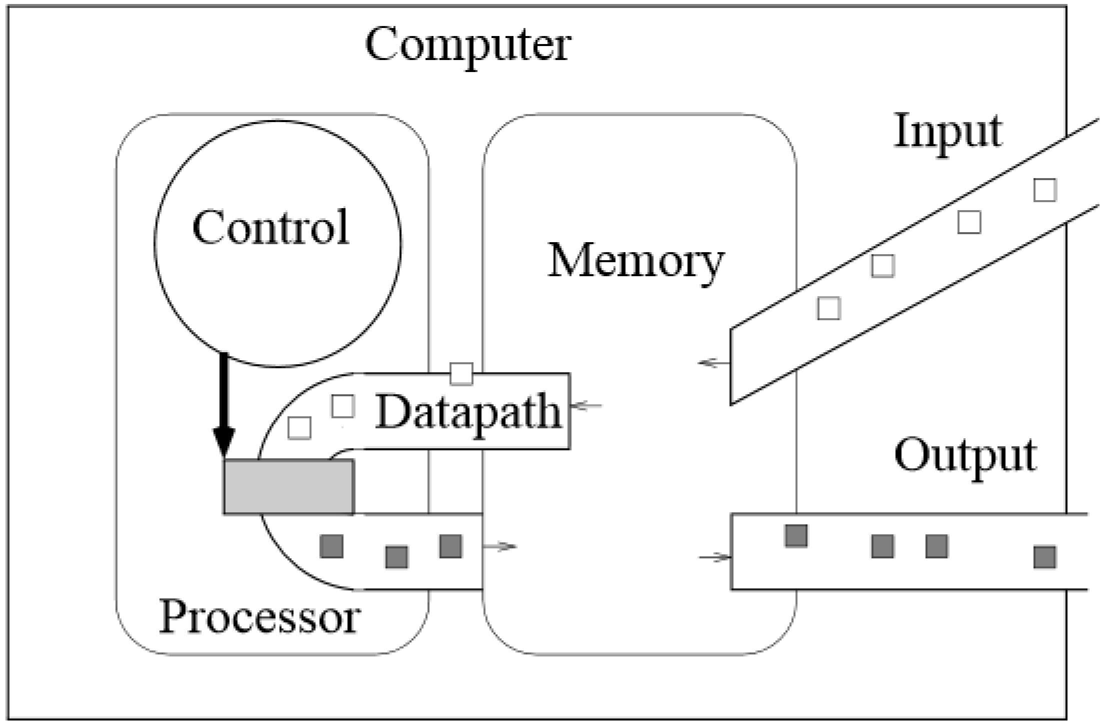
- This will better enhance performance rather than optimizing the rare case. (common case is usually much simpler to optimize)

Guiding Principles : Computer Architecture Design

- **Improve Performance Via Parallelism**
  - Doing multiple tasks at once.
  - Divide and Conquer
- **Improve Performance Via Pipelining**
  - Prevalent in computer architecture design
- **Improve Performance Via Prediction**

  can be better to ask for forgiveness later than to ask for permission

Big Picture

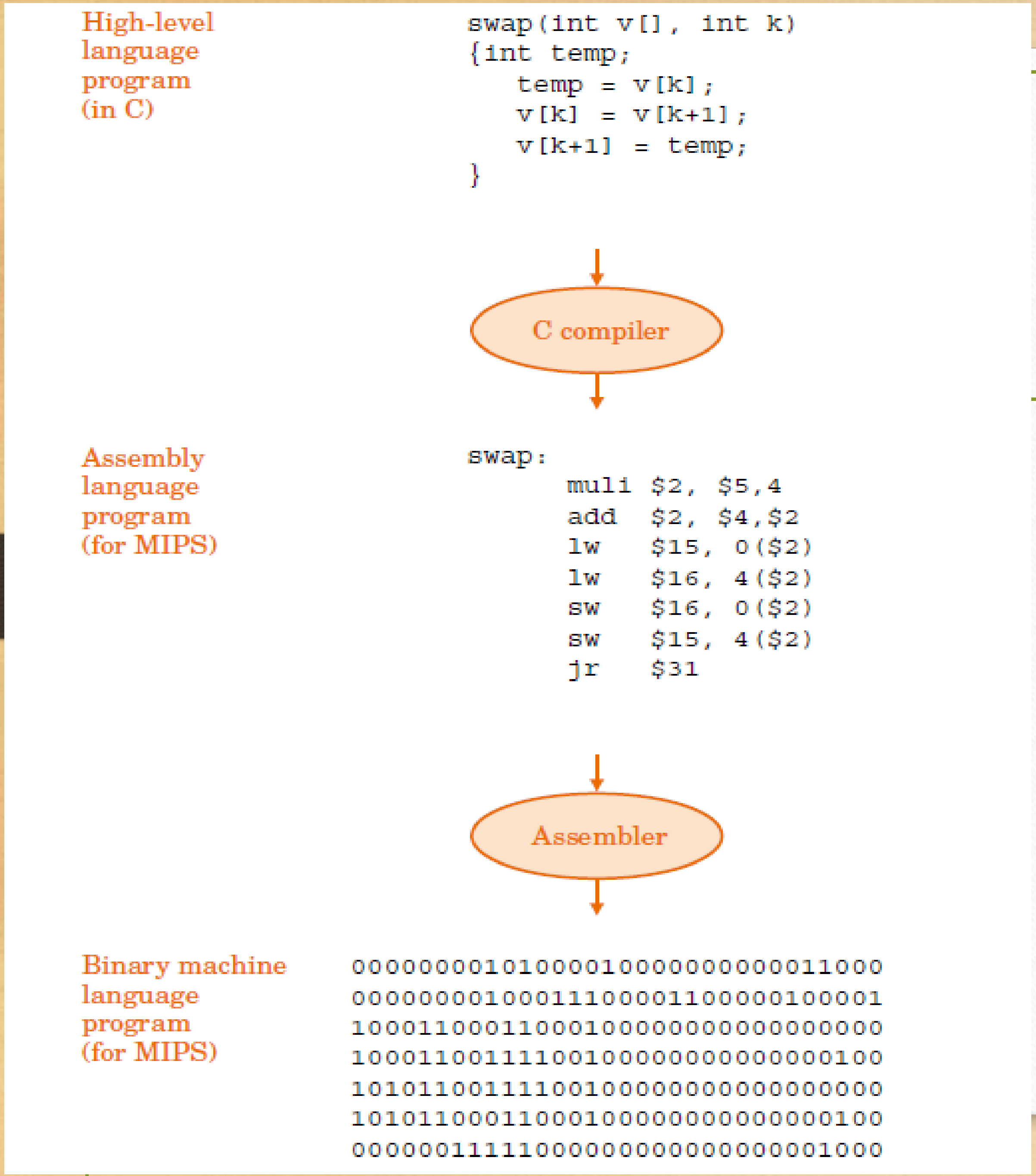


Instruction Set Architecture

- **To connect to the Hardware, you must speak its language**
  - Machine language
  - Send it machine level instructions to interpret and execute
- Different computers speak different dialects of the same language
  - MIPS: Instruction set from MIPS technologies 1980s
  - ARM : Similar to MIPS , iPad A4
  - Intel x86 Base architecture which extends the PC and much of PostPC era

Similarities in all these Architectures

- **There must be many similarities in these Instruction sets**
    - Hardware underlying the systems are based on the same technological principles
    - There are common basic operations that all computers must be able to do. Add, subtract, perform arithmetic operations
    - Goals are similar: Maximize performance and minimize costs and energy


High-level language program (in C)
C compiler
Assembly language program (for MIPS)
Assembler
Binary machine language program (for MIPS)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

```
00000000101000010000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

High Level Languages

Compiled or Interpreted
Eventually into Bytecode

These Instructions Execute in the Pipeline
The hardware in the pipeline interprets the Instruction piece by piece (bit by bit) ☺

Assembly Language such as MIPS:

- A**ssembly language** :low-**level** programming **language** for a computer
- Assembly language instructions translated into Machine code :
- Executed on the lower level machine : Processor.
- Ordinary complied code run 'directly' on the CPU , but programs do not run in vacuum. OS plays a critical role (system calls, access to I/O, DLL)
- VM between compiled executable and underlying instruction set architecture of the processor
- Interpreted programs do not worry about the quirks and differences between ISA
- x86 standard calling conventions

MIPS

- **What does MIPS stand for?**

MIPS

- **Millions of Instructions per Second**

MIPS

- **Millions of Instructions per Second:**
- **Yes but that's not the MIPS we are interested in right now**

MIPS

- **MIPS:**
- **Microprocessor without Interlocked Pipelined Stages….**
- **RISC architecture : Reduced Instruction Set**
  - **Very Basic operations**
  - **We look at Base MIPS 32 bit**

  More advanced versions of MIPS developed later

  ie) 64 bit etc.

MIPS

refer to course notes and a0

- **C code:**
  - **f = (g + h) – (i + j)**

Assume that variables f , g h and i and j are assigned to

Registers \$s1….\$s5 respectively

MIPS

- C code:
  - $f = (g + h) - (i + j)$

Assume that variables f , g h and i and j are assigned to

Registers $s1….$s5 respectively

Use two temp registers $s6 and $s7

- $s1: f $s2: g $s3: h $s4: i $s5: j

f= (g +h)- (i +j)

MIPS:

```
add $s6, $s2, $s3
add $s7, $s4, $s5
```

- $s1: f $s2: g $s3: h $s4: i $s5: j

f= (g +h)- (i +j)

MIPS:

```
add $s6, $s2, $s3

add $s7, $s4, $s5

sub $s1, $s6, $s7
```

Performance

- Measuring performance :
  - Depends on how you think about it
- Running the same program on two different desktops
  - Faster one gets the job done first
- Datacenter : Several servers running inquiries submitted by
  - Many users
  - We would like to know: How many completed user tasks per day
  - Or completed jobs

Performance

- Individual Computer
  - Response Time: Time between start and completion of a task
    - Also known as *Execution Time*
- Datacenter Manager:
  - Throughput: Total amount of work done in a given time

Execution Time/ Throughput

Would the following improvements improve the Execution time, Throughput or both:

- Changing the processor to a newer and faster version
- Adding in additional processors to a system, thereby allowing multiple tasks to begin execution. (each processor is dedicated to an individual task)

Execution Time/ Throughput

Would the following improvements improve the Execution time, Throughput or both:

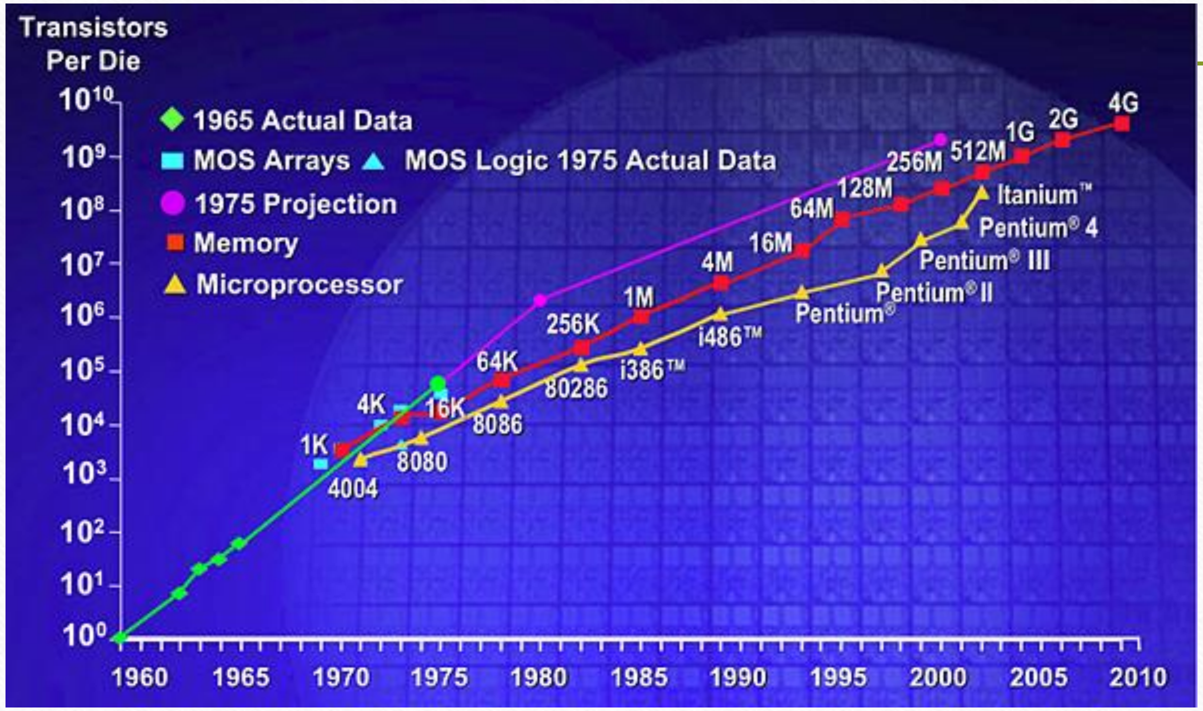
- Changing the processor to a newer and faster version

 *decrease in execution time and increase in throughput: More tasks able to complete

- Adding in additional processors to a system, thereby allowing multiple tasks to begin execution. (each processor is dedicated to an individual task)
- execution time of individual tasks remains the same.
- Throughput increase: More tasks completed in a given amount of time

Uniprocessors to Multiprocessors

"Moore's Law" (Moore's Trend) describes the tendency for the number of transistors that can be place on an integrated circuit board to double approximately every two years.



Design Gap:
Are we able to maximize the use
Of these advents in technology

Leveling off:
Have we now reached or come close to reaching a Plateau.

Power Consumption : Improvements are not
At the same rate

Image Source: http://business-technology.co.uk/2011/10/mastering-the-art-of-electronics-evolution/

Multiprocessor Cores

- Rather than continuing to decrease the response time of single program running on a single microprocessor → multiple processors were added per chip.
- Now the internal config of a microprocessor referred to as a Core, and such multiprocessor chips described as multicore microprocessors

Parallelism

- In the past programmers could rely on improvements in hardware or in compilers to double the performance of their program
- Now= Programmers will need to start re-thinking rewriting their programs to take advantage of hardware.
- Multicore microprocessors: Parallel Computing
- Parallelism in the Pipeline : Discussed in Pipelining Unit