

---

## CIS42 - Evergreen Valley College Introduction to Modern Structured Programming

Lecture: Room AC162  
Labs: Room RG240

### Lecture 1

Class Home Page: <http://www.engr.sjsu.edu/~cpham/>

CIS42-EVC Lec#1 Intro.1

Instructor: Christopher H. Pham  
<http://www.engr.sjsu.edu/~cpham/>

Some material adapted from Patterson@UCB

---

## Overview

- Intro to Computer Architecture & Hardware/Software Interface (20 min)
- Modern Structure Programming (30 min)
- Course Style, Philosophy and Structure (10 min)
- Administrative Matters (15 min)
- Break (15 min)
- Programming & Problem-Solving (30 min)
- Introduction to C++ (30 min)
- Testing & Debugging (20 min)

CIS42-EVC Lec#1 Intro.2

Instructor: Christopher H. Pham  
<http://www.engr.sjsu.edu/~cpham/>

Some material adapted from Patterson@UCB

---

## What is “Computer Architecture”

**Computer Architecture =  
Instruction Set Architecture +  
Machine Organization**

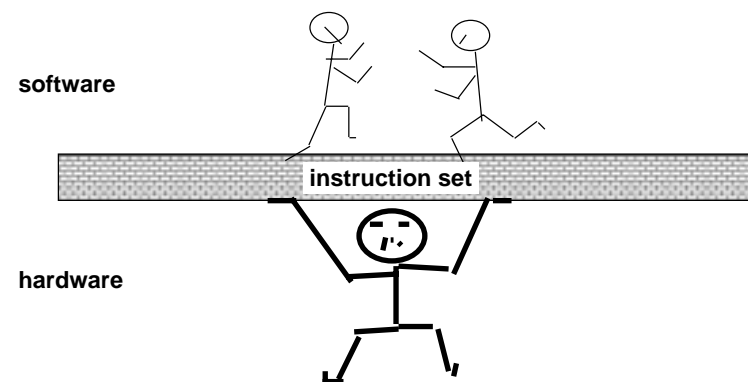
CIS42-EVC Lec#1 Intro.3

Instructor: Christopher H. Pham  
<http://www.engr.sjsu.edu/~cpham/>

Some material adapted from Patterson@UCB

---

## The Instruction Set: a Critical Interface



CIS42-EVC Lec#1 Intro.4

Instructor: Christopher H. Pham  
<http://www.engr.sjsu.edu/~cpham/>

Some material adapted from Patterson@UCB

## Example ISAs (Instruction Set Architectures)

◦ Digital Alpha	(v1, v3)	1992-97
◦ HP PA-RISC	(v1.1, v2.0)	1986-96
◦ Sun Sparc	(v8, v9)	1987-95
◦ SGI MIPS	(MIPS I, II, III, IV, V)	1986-96
◦ Intel	(8086, 80286, 80386, 80486, Pentium, MMX, ...)	1978-96

## Organization

### ◦ Capabilities & Performance Characteristics of Principal Functional Units

- (e.g., Registers, ALU, Shifters, Logic Units, ...)

### ◦ Ways in which these components are interconnected

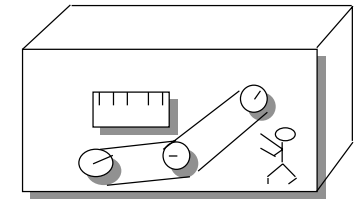
### ◦ Information flows between components

### ◦ Logic and means by which such information flow is controlled.

*Logic Designer's View*

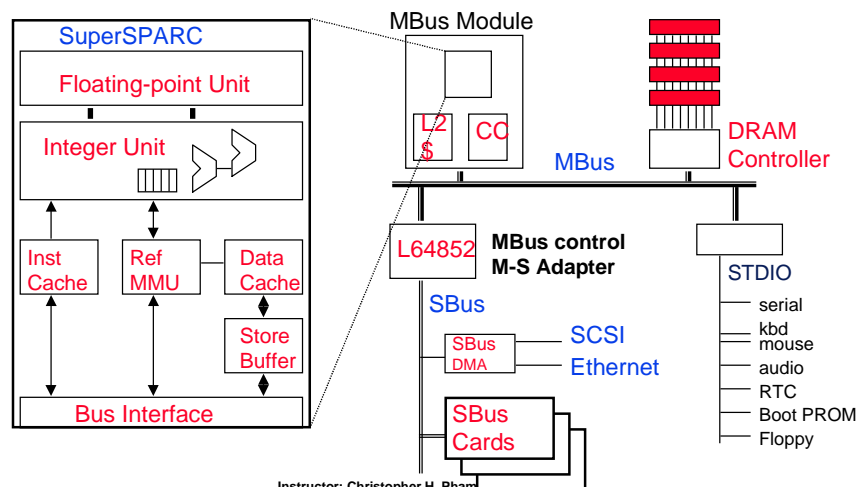
ISA Level

FUs & Interconnect

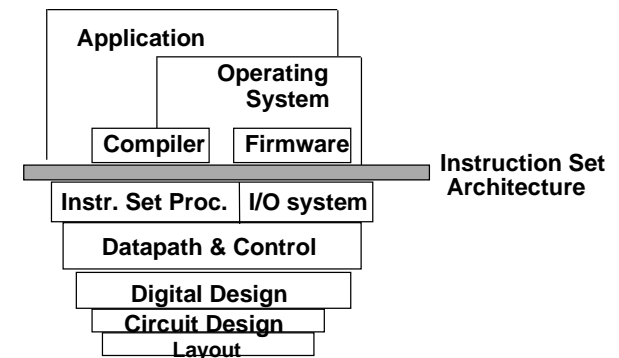


## Example Organization

### ◦ TI SuperSPARC™ TMS390Z50 in Sun SPARCstation20



## What is "Computer Architecture"?

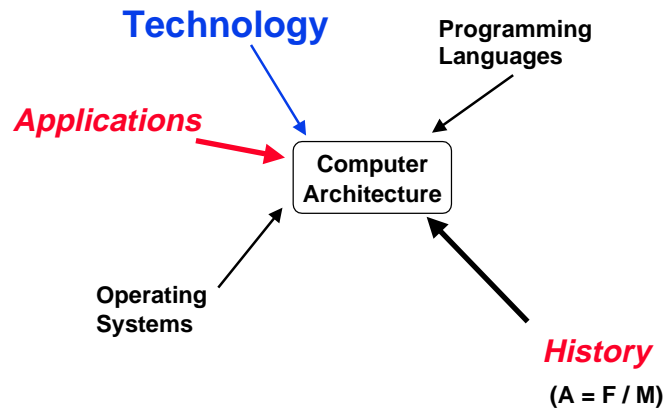


### ◦ Coordination of many *levels of abstraction*

### ◦ Under a rapidly *changing set of forces*

### ◦ Design, Measurement, *and* Evaluation

## Forces on Computer Architecture



## Technology => dramatic change

- **Processor**
  - logic capacity: about 30% per year
  - clock rate: about 20% per year
- **Memory**
  - DRAM capacity: about 60% per year (4x every 3 years)
  - Memory speed: about 10% per year
  - Cost per bit: improves about 25% per year
- **Disk**
  - capacity: about 60% per year

## Applications and Languages

- CAD, CAM, CAE, . . .
- Lotus, DOS, . . .
- Multimedia, . . .
- The Web, . . .
- C, C++, JAVA, . . .
- ???

## CIS42: Course Content

### Introduction to Modern Structured Programming

- |                     |                  |
|---------------------|------------------|
| Problem Solving     | Algorithm        |
| C++ as a tool       | Systems Overview |
| Testing/Debugging   | Software Tools   |
| Software Life Cycle | Other Techniques |

## CIS42: So what's in it for me?

---

- In-depth understanding of the inner-workings of modern structured programming, software evolution, problem-solving techniques, trade-offs present at the hardware/software boundary, software tools and testing/debugging techniques.
- Experience with the *design process* in the context of a simple to medium complex (software) design.
  - Spec --> Algorithm --> High level implementation -> Coding
  - Modern software programming tools

## Typical Lecture Format

---

- 35-Minute Lecture
- 5 to 10-Minute Administrative Matters
- 30-Minute Lecture
- 15-Minute Break (snack, water, stretch) (?)
- 25-Minute Lecture
- 5-Minute Break
- 25-Minute Lecture
- 5-Minute Break
- 20-Minute Lecture
- Instructor will stay after to answer questions

## Course Administration

---

- Instructor: Christopher H. Pham
- Lecture: Room: AC162
- Labs: Open labs available. Room: RG240
- Materials: <http://www-engr.sjsu.edu/~cpham/>
- Text: Problem Solving with C++, by Walter Savitch, 2nd Edition.

## Course Exams

---

- Reduce the pressure of taking exams
  - All exams & finals will be curved. See Green Sheet for grade/score translation.
  - Both mid-term/final exams can bring summary sheets

## Course Workload

---

- Reasonable workload (if you have good work habits)
  - Simplified projects & final project
  - Project teams may have 2 members

## Homework Assignments and Project

---

- Most assignment consists of two parts
  - Individual Effort: Exercises from the text book, includes the Self-Test Exercises at the end of each chapter.
  - Team Effort: Proj. assignments
- Assignments
  - Exercises & projects due at beginning of discussion section
- Projects/Homeworks returned in discussion section
- Must turn in survey to be considered enrolled

## My Goal

---

- Show you ...
- NOT to talk at you
- SO...
  - ask questions
  - find me in the lab
  - send me emails
  - ...

## Course Problems

---

- Can't make midterm
  - Tell early us and we will schedule alternate time
- Forgot to turn in homework/ Dog ate computer
  - As a result of feedback, going to grade almost immediately so that can give results back quickly => late homeworks a hassle
- What is cheating?
  - Studying together in groups is encouraged
  - Work must be your own
  - Common examples of cheating: running out of time on a assignment and then pick up output, take homework from box and copy, person asks to borrow solution "just to take a look", copying an exam question, ...
  - Better off to skip assignment (Projs + homeworks 40% of grade)

## Class decides on penalties for cheating; staff enforces

---

- **Exercises (book):**
  - 0 for problem
  - 0 for homework assignment
  - subtract full value for assignment
  - subtract 2X full value for assignment
- **Labs leading to project (groups: only penalize individuals?)**
  - 0 for problem
  - 0 for homework assignment
  - subtract full value for assignment
  - subtract 2X full value for assignment
- **Exams**
  - 0 for problem
  - 0 for exam

## Project Simulates Industrial Environment

---

- **Communicate with colleagues (team members)**
  - What have you done?
  - What answers you need from others?
  - You must document your work!!!
  - Everyone must keep an on-line notebook
- **Communicate with supervisor (Instructor)**
  - How is the team's plan?
  - Short progress reports are required:
    - What is the team's game plan?
    - What is each member's responsibility?

## Things We Hope You Will Learn from CIS42

---

- **Keep it simple and make it work**
  - Fully test everything individually and then together
  - Retest everything whenever you make any changes
  - Last minute changes are big "no nos"
- **Group dynamics. Communication is the key to success:**
  - Be open with others of your expectations and your problems
  - Everybody should be there on design meetings when key decisions are made and jobs are assigned
- **Planning is very important:**
  - Promise what you can deliver; deliver more you promise
  - Murphy's Law: things *DO* break at the last minute
    - Don't make your plan based on the best case scenarios
    - Freeze you design and don't make last minute changes
- **Never give up! It is not over until you give up.**

## What you should know from prerequisites?

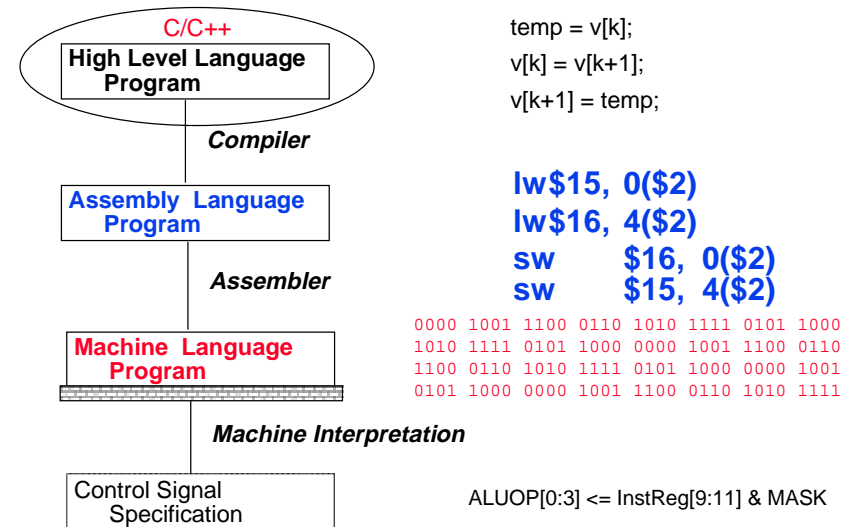
---

- English
- Basic computer skills: typing, printing, emails, www, etc.

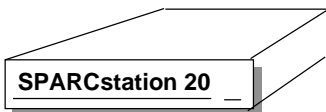
## Getting into CIS42

- If not pre-enrolled, Fill out Add Form
- Fill out survey and return after class
- **Know the prerequisites**
  - CIS41 - Introduction to CIS

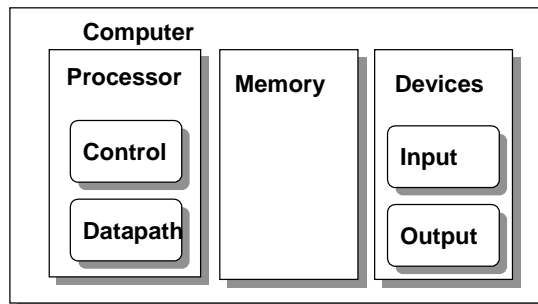
## Levels of Representation (Overview)



## Levels of Organization



Workstation Design Target:  
25% of cost on Processor  
25% of cost on Memory  
(minimum memory size)  
Rest on I/O devices,  
power supplies, box



## Compiler - Linker

- A **COMPILER** is a program that translates a high-level language program into a machine-language program called **OBJECT CODE**.
- A **LINKER** combines different object codes. The process is called **LINKING**.

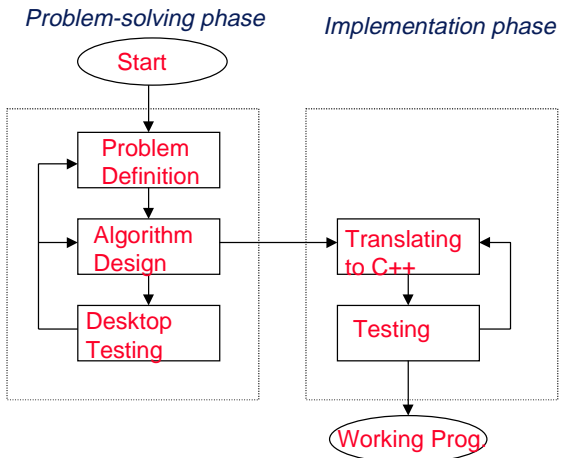
## Programming and Problem-Solving (1)

◦ **Algorithm: is a sequence of precise instruction that leads to a solution.**

- Text, page 15: Algorithm to count the number of same name
  - 1. Get the list of names
  - 2. Get the name being checked
  - 3. Set counter to zero
  - 4. For each name on the list:
    - compare each name on the list to the checked name
    - if same name found, add one to counter
  - 4. Output the final counter value

## Programming and Problem-Solving (2)

◦ **Program Design (outline)**



## Programming and Problem-Solving (3)

◦ **The Software Life Cycle (6 phases)**

1. *Problem Definition*: Analysis & Specification of the task
2. *Algorithm Design*: Design of the SW
3. *Implementation*: coding
4. *Testing*
5. *Maintenance & evolution* of the system
6. *Obsolescence*

## Introduction to C++

- BCPL -> B Language (not our concern)
- C Language (Dennis Ritchie - AT&T Bell Labs 1970s)
- C++: (Bjarne Stroustrup - AT&T Bell Labs)
  - Most C is a subset of C++ ( NOT vice versa though !!!)
  - Unlike C, C++ can do *object-oriented* programming.

## A Sample C++ Program (1)

---

```
#include <iostream.h>

int main() {

    variable_declarations

    statement_1
    statement_2
    ...
    statement_n

    return 0;

}
```

## A Sample C++ Program (2)

---

```
#include <iostream.h>

int main() {
    int number_of_dimes, amount_in_dollars;
    cout << "Press return after entering a number.\n";
    cout << "Enter number of dimes:\n";
    cin >> number_of_dimes;
    //there are 10 dimes in a dollar
    amount_in_dollars = number_of_dimes * 10;
    cout << "For number of dimes = ";
    cout << number_of_dimes;
    cout << "\nTotal amount in dollars = $"
    cout << amount_in_dollars;
    cout << "\n";
    return 0;
}
```

## A Sample C++ Program (3)

---

◦ Sample output from previous C++ program:

Press return after entering a number.

Enter number of dimes:

100

For number of dimes = 10

Total amount in dollars = \$10

Note: we have not covered the decimal points yet.

## Compiling and Running a C++ Program

---

- Write a program using any Text Editor
- If you use a word processor, save as text.
- Will need to compile, link and run the program
- May need to create a project, depends on the tool
  
- MORE IN LAB

## Testing and Debugging

---

- **Syntax Errors:** will be caught by the Compiler (hopefully)
- **Error Messages vs. Warning Messages**
- **Run-time Errors:** only detected when the program is actually run.
- **Logic Errors:** mistakes in the algorithm or in the translation of the algorithm to C++.