[Skip To Content](https://clemson.instructure.com/courses/59838/pages/study-guide-for-exam-1-and-sample-exams?module_item_id=840928" \l "content)

[Dashboard](https://clemson.instructure.com/)

[Account](https://clemson.instructure.com/profile)

[Dashboard](https://clemson.instructure.com/)

[Courses](https://clemson.instructure.com/courses)

[Groups](https://clemson.instructure.com/groups)

[Calendar](https://clemson.instructure.com/calendar)

[15](https://clemson.instructure.com/conversations)

[Inbox](https://clemson.instructure.com/conversations)

[Help](http://help.instructure.com/)

* [S1808-CPSC-2310-002](https://clemson.instructure.com/courses/59838)
* [Pages](https://clemson.instructure.com/courses/59838/pages)
* Study Guide for Exam 1 and Sample Exams

*Fall 2018*

* [Home](https://clemson.instructure.com/courses/59838)
* [Announcements](https://clemson.instructure.com/courses/59838/announcements)
* [Assignments](https://clemson.instructure.com/courses/59838/assignments)
* [Discussions](https://clemson.instructure.com/courses/59838/discussion_topics)
* [Grades**1**](https://clemson.instructure.com/courses/59838/grades)
* [People](https://clemson.instructure.com/courses/59838/users)
* [**Pages**](https://clemson.instructure.com/courses/59838/wiki)
* [Files](https://clemson.instructure.com/courses/59838/files)
* [Syllabus](https://clemson.instructure.com/courses/59838/assignments/syllabus)
* [Modules](https://clemson.instructure.com/courses/59838/modules)
* [Conferences](https://clemson.instructure.com/courses/59838/conferences)
* [Collaborations](https://clemson.instructure.com/courses/59838/collaborations)

1. Be able to briefly define, describe, or explain the following terms:

        ALU (arithmetic/logic unit)     registers  
        memory                          CPU (central processing unit)  
        machine language                assembly language  
        PC (program counter)            von Neumann cycle (fetch/execute cycle)  
        stack machine                   accumulator machine  
        load/store machine              assembler  
        label                           compiler   
        symbol table                    location counter

ALU: Executed operations

Memory: Stores data to be quickly accessed

Machine language: Code stored in bytes for processor to read

Program Counter: Points to next instruction

Stack machine: Processor that uses a stack to store instructions and values

Load/store machine: Processor that uses registers to store instructions and values

Label: Codenamed assosciated for an address that an assembler uses to be pointed to

Symbol table: Store of symbols and assosciated values or addresses

Registers: Very fast shortage

CPU: Machine that executes programs

Assembly language: A language that assembler converts to machine code

Von Neumann cycle:

Accumulator machine: Simple CPU that uses accumulator to operate on values

Assembler: Converts assembly language to machine code

Compiler: Converts high level language to assembly code

Location counter: increments address locations depending on operation size

Also be able to briefly define, describe, or explain these additional terms:

        address                         MAR (memory address register)  
        MDR (memory data register)      IR (instruction register)  
        opcode                          mnemonics  
        object code                     instruction  
        load                            store  
        branch                          pseudo-op  
  
        direct addressing               immediate addressing  
        indirect addressing             register addressing

Address: Location of value in memory

MDR: Register that contains data to be stored in or feteched from memory

Opcode: Code for an instruction

Object code: Code for an object that holds data

Load: Take from memory

Branch: Shift through assembly code

Direct Addressing: Addressing where operand address is explicitly given in code

Indirect addressing: Addressing where pointer to operand is given in code

MAR: CPU Stores addresses from which data will be stored or fetched

IR: Holds current instruction

Mnemonics:

Instruction: Piece of code that tells computer how to manipulate data:

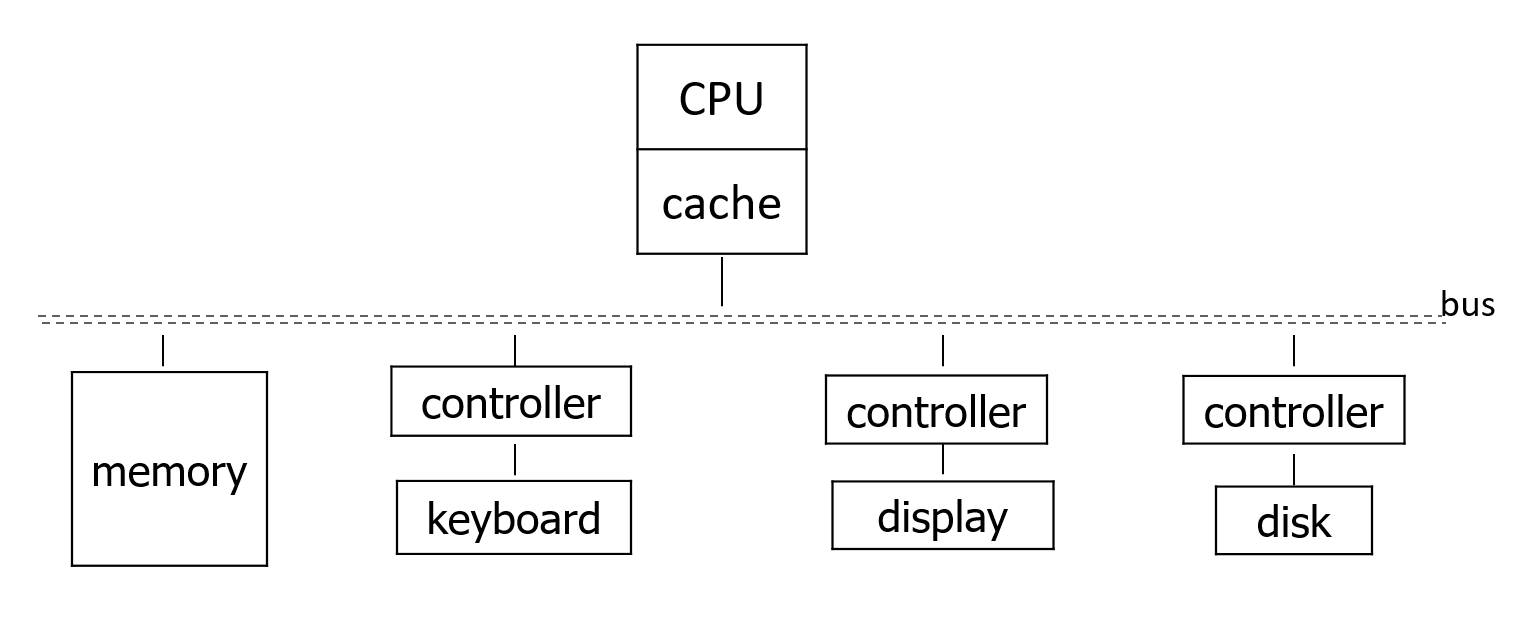
Store: Put data into memory or storage

Pseudo-op: Instruction-like statement that is simply to help the assembler

Immediate addressing: Addressing where operand is given in code

Register address: Addressing where register containing operand is given in code

1. Be able to draw a diagram of a simple computer system and label these parts: CPU, cache, bus, memory, I/O controller, I/O device.



1. Know the power of ten that corresponds to given prefixes, such as

        (K) kilo-                       (m) milli-  
        (M) mega-                       (greek mu) micro-  
        (G) giga-                       (n) nano-  
        (T) tera-                       (p) pico-  
        (P) peta-

1. Be able to explain **why** forward references require a two-pass structure for a translator.

Translator would otherwise have no way to know locations of said forward referencing

1. Be able to explain and/or demonstrate **what** each pass does for a two-pass assembler.
2. Given a math expression or HLL code segment, be able to write the equivalent assembly language code for the accumulator machine.

Example hand assembly  
  
   The opcodes for the load, sub, store, and halt instructions are 50, 30, 60, and 0, respectively.

   program              symbol table      executable  
  
     word(a,5)              a  0             5  
     word(b,a)              b  1             0  
   label(x)                 x  2  
   label(y)                 y  2  
     load(a)                                 50  0  
     store(b)                                60  1  
     halt                                    0  
   end(x)                                    2  
  
  
   program              symbol table      executable  
  
   label(start)           start  0  
     load(ten)                               50  9  
     sub(ten)                                30  9  
     store(data)                             60  7  
     halt                                    0  
     word(data,five)      data   7           8  
     word(five,5)         five   8           5  
     word(ten,10)         ten    9           10  
   end(start)