

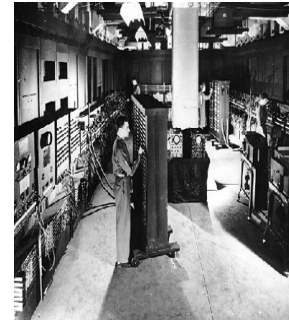
Chapter 4

The Von Neumann Model

Based on slides © McGraw-Hill
Additional material © 2004/2005 Lewis/Martin
Modified by Diana Palsetia

How does the computer get the instructions?

- Before stored program concept
 - plugboards
- Stored program concept
 - the program is stored in memory and computer then will get the instructions one-by-one and process them.
 - This eliminated physical feeding of instructions
- Proposed by J. Mauchly & J. Eckert
- Popularized by Von Nuemann



Copyright: ENIAC

CIT 593

4-3

How does the computer work?

To perform a task, the computer is provided with **program**

Program

- Consists of a **set of instructions** that the computer must do to complete the task
- e.g. Procedure to obtain a Driver's license

Instruction

- Smallest piece of work specified (fundamental unit)
- Either carried completely or not at all (this is because future instructions can be dependent on the previous)
- e.g. Obtain your permit before you take the driving test

CIT 593

4-2

Von Nuemann Model(stored program concept)

Von Neumann Machine (or Model) consists of the following components to process a program:

- **Memory**: where instructions and data are stored
- **Control unit**: co-ordinates all other units. It also interprets instructions
- **Processing unit**: for performing arithmetic and logical operations
- **Input/Output units**: for interacting with *real world*

CIT 593

4-4

Computer vs. Human Brain

Consider the case where someone is being told a story which they then have to repeat to someone else.

The person being told the story will have to:

- **listen** to the story
- **process** i.e. pay attention to the story so that it goes from the ears to being understood
- **put it in memory** until there is a need to tell it again
- **repeat** i.e. tell it again

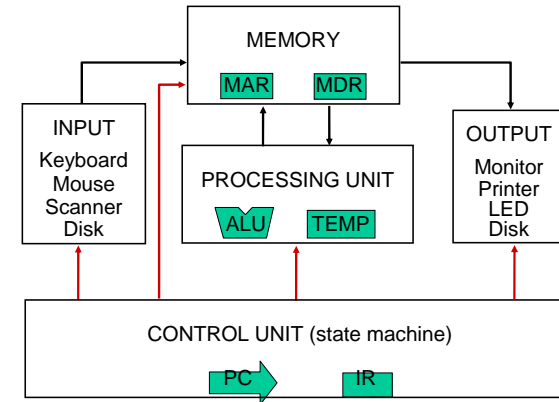
Note: The brain is the controller for the above activities

CIT 593

Example copyright: University of Westminster, UK

4-5

Von Neumann Model



CIT 593

4-7

Computer vs. Human Brain (contd..)

For a computer these are done as follows:

Listen: use the input devices, such as the keyboard or a microphone to 'receive' the story.

Processing: do the thinking in the Central Processing Unit (CPU) of the computer.

Put in memory: put the story in the computer's memory.

Repeat: use the output devices, such as the monitor or the printer to 'tell' the story to a person using the computer.

Control Unit: analogous to brain, which controls the above activities

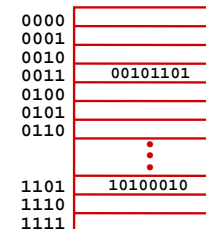
CIT 593

4-6

Memory

Basic Operations

- **Load(LD):** read a value from a memory location
- **Store(ST):** write a value to a memory location



CIT 593

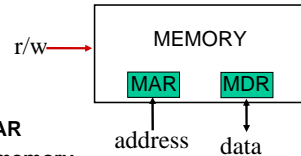
4-8

Interface to Memory

How does processing unit get data to/from memory?

MAR: Memory Address Register

MDR: Memory Data Register



To read a location A

1. Write the address A into the MAR
2. Send a “read (r)” signal to the memory
3. Read the data from MDR

To write a value X to a location A

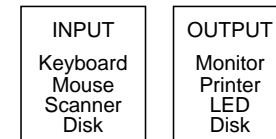
1. Write the data X to the MDR
2. Write the address A into the MAR
3. Send a “write (w)” signal to the memory

CIT 593

4-9

Input and Output (I/O)

Devices get data into and out of computer



Each device has own interface

- Often a set of registers like the memory’s MAR and MDR
- LC-3 supports keyboard (input) and display (output)

Some devices provide both input and output

- E.g. Disk

Software that controls device access

- *Driver*

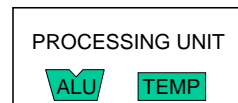
CIT 593

4-11

Processing Unit

Functional Units

- ALU = Arithmetic and Logic Unit
- Could have many functional units (some special-purpose, e.g., multiply, square root, ...)
- LC-3: ADD, AND, NOT



Registers

- Small, temporary storage
- Operands(data to be operated on) and results of functional units
- LC-3: eight register (R0, ..., R7)

Word Size

- Number of bits normally processed by ALU in one instruction
- Also width of registers
- LC-3: 16 bits

CIT 593

4-10

Control Unit

Orchestrates execution of the program (like your Brain)

- Keeps track of where we are in the process of executing >the program as well as each instruction

Instruction Register (IR)

- Contains the current instruction

Program Counter (PC)

- Contains the address of the next instruction to execute



Control Unit (Giant State Machine)

- Reads an instruction from memory (at PC)
- Interprets the instruction
- Generates signals that tell the other components what to do
- Instruction may take many *machine cycles* to complete

CIT 593

4-12

Instructions

Fundamental unit of work

Constituents

- **Opcode:** operation to be performed (e.g. add, and)
- **Operands:** data/locations to be used for operation
 - **Source:** location that contains the data
 - **Destination:** location that will store the result of computation
 - **Immediate:** data values not contained at a particular location

Encoded as a sequence of bits (*just like data!*)

- Sometimes have a fixed length (e.g., 16 or 32 bits)
- Control unit interprets instruction
 - Generates control signals to carry out operation
 - **Atomic:** operation is either executed completely, or not at all

CIT 593

4-13

ISA

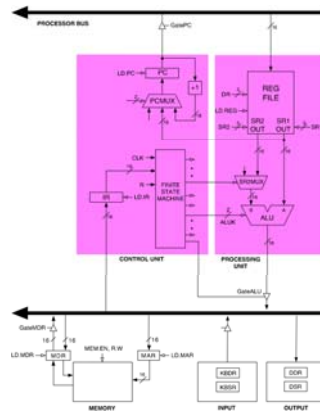
Instruction Set Architecture (ISA)

- Computer's instructions
 - ADD, NOT, SUB
- Instruction Format
 - Which bits are opcodes, which are operands
- Instruction Behaviour
 - How instructions perform
- Other
 - addressing modes, memory architecture, interrupt and exception handling, and external I/O (more details in Chp5)

CIT 593

4-15

Example of von Nuemann Model: LC3



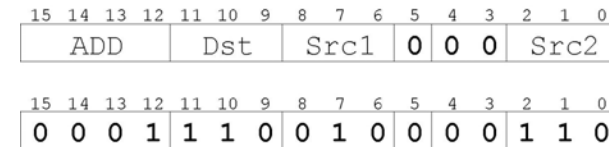
CIT 593

4-14

Example

LC-3 has 16-bit instructions

- Each instruction has a **four-bit opcode, bits [15:12]**
- Has **eight registers (R0-R7)** for temporary storage



LC-3 ADD : Add the contents of R2 to the contents of R6, and store the result in R6.

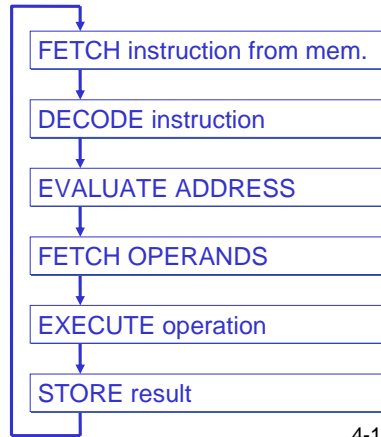
CIT 593

4-16

Instruction Processing

Question

- How are instructions executed?



CIT 593

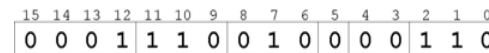
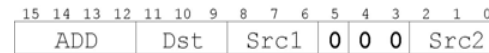
4-17

Instruction Processing: DECODE

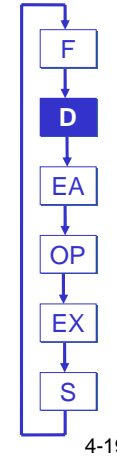
Identify opcode

- Using the opcode bits
 - In LC-3, always first four bits of instruction

Identify operands from the remaining bits



Control unit implements DECODE



CIT 593

4-19

Instruction Processing: FETCH

Idea

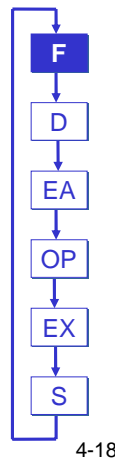
- Put next instruction in IR & increment PC

Steps

- Load contents of PC into MAR
- Increment PC
- Send "read" signal to memory
- Read contents of MDR, store in IR

Who makes all this happen?

- Control unit



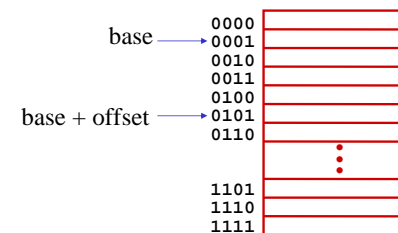
CIT 593

4-18

Instruction Processing: EVALUATE ADDRESS

Compute address

- For loads (reading to mem) and stores (writing to mem)
- Addressing is not straightforward
 - Reference mem address is known (base)
 - If you want to go to the 4th location from base then offset is set to 4



CIT 593

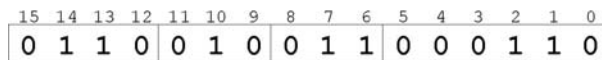
4-20

Example: LC-3 LDR Instruction

Reads data from memory

Base + offset addressing mode

- Add offset to base register to produce memory address
- Load from memory address into destination register



“Add the value 6 to the contents of R3 to form a memory address. Load the contents of memory at that address and place the resulting data in R2.”

CIT 593

4-21

Instruction Processing: EXECUTE

Actually perform operation

Examples

- Send operands to ALU and assert ADD perform operation based on opcode (e.g. ADD, AND)
- Do nothing (e.g., for loads(reads) and stores(writes))



CIT 593

4-23

Instruction Processing: OPERAND fetch

Get source operands for operation

Examples

- Read data from register file (e.g. for an ADD)



CIT 593

4-22

Instruction Processing: STORE

Write results to destination

- Register or memory

Example

- Result of ADD is placed in destination reg.
- If we have limited space
 - Some values from registers are written back to memory via STORE command

Note: The store command will set MDR and assert WRITE signal to memory



CIT 593

4-24

Changing the Sequence of Instructions

Increment of PC

- FETCH phase always increments PC by 1 instruction
- But we can also increment PC by the number of instructions we skipped in certain conditions

We can also skip instructions:

- programming constructs that change the sequence of instruction execution like *If-then*, *loops* etc.
 - E.g: *If (Overall points equals 90) then Grade equals A*

Instructions that change PC other than +1 in assembly:

- **Branches** are conditional
 - Change the PC only if some condition is true e.g., the contents of a register is zero
- **Jumps** are unconditional -> Always change the PC

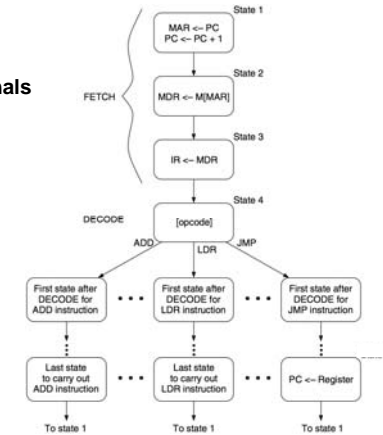
CIT 593

4-25

Control Unit Details

Finite state machine

- Input: PC, IR
- Output: *many* control signals
E.g., MAR ← PC
⇒ GatePC and LD.MAR

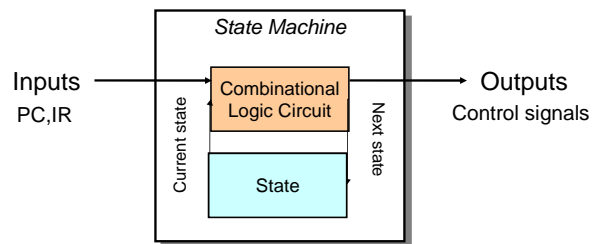


CIT 593

4-27

How Does Control Unit Work?

Remember state machines ?

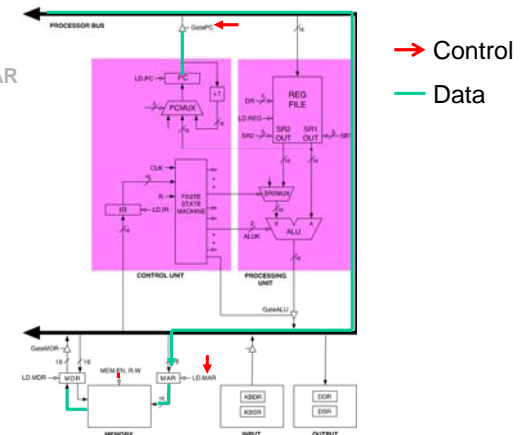


CIT 593

4-26

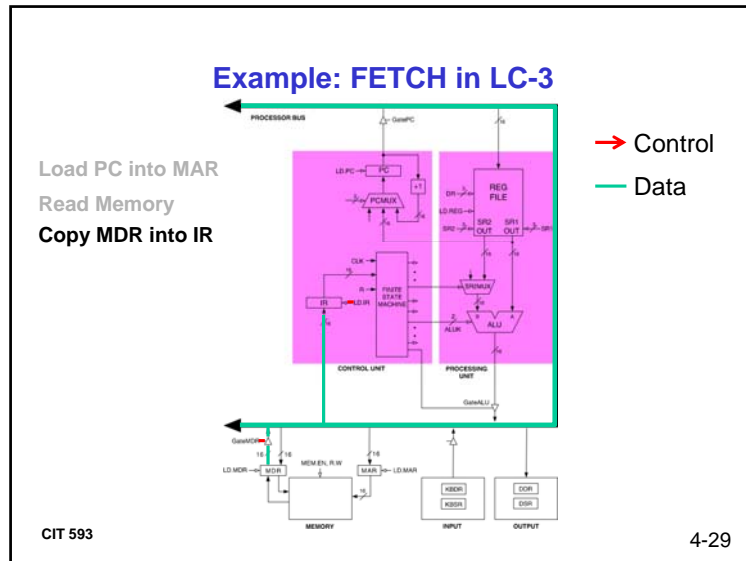
Example: FETCH in LC-3

Load PC into MAR
Read Memory



CIT 593

4-28



Instruction Processing Summary

Instructions look just like data

- Interpreted by machine

Three basic kinds of instructions

- Computational instructions (ADD, AND, ...)
- Data movement instructions (LD(load), ST(store), ...)
- Control instructions (JMP, BRnz, ...)

Six basic phases of instruction processing

F → D → EA → OP → EX → S

- Not all phases are needed by every instruction
- Phases may take variable number of machine cycles
- Multiple phases per cycle possible

CIT 593 4-31

Stopping the Computer

How does the computer know its done after x number instructions?

- e.g. $A * B + C$ – 3 loads + 1 MULT + 1 ADD = 5 total instructions

Older Machines

- Done by a **HALT** instruction

Modern machines

- User programs execute under the control of O.S.
 - Once the program terminates, the control instruction changes PC to again start O.S.

CIT 593 4-30

Next Time

Lecture

- LC-3 Chapter 5

Reading

- Chapter

Upcoming

- Homework 1 due 9/21

CIT 593 4-32