EE2004

Prof. C.S. Leung
Room: G6520
Phone: 34427378
Email: eeleungc@cityu.edu.hk

Text Book:
PIC Microcontroller: An Introduction to Software & Hardware Interfacing, Han-Way Hunag

PIC Microcontroller and Embedded Systems: Using Assembly and C for PIC18

Reference:
Computer Organization, Carl Hamacher, Zvonko Vranesic, Safwat Zaky
Mc-Graw Hill

You can download some software packages from
http://www.microchip.com/development-tools/downloads-archive

Note : you should select **MPLAB IDE v8.XX versions rather than MPLAB IDE X.**

Examination 60% Continuous Assessment 40%

Tutorial Exercise, Quizzes and tests (25 %)

**At least 75% tutorial attendance rate must be obtained.
Otherwise, you get zero mark in the continuous assessment.**

Mini Project **15 % and (**10 % extra bonus)

2 students form a group for tutorial exercise and mini project

To pass the course,
at least 35 marks (out of 100) in the examination,
at least 35 marks (out of 100) in CA,

Successful completion of this course, students should be able to

1. Describe the structure and major components of a microcomputer system.
2. Describe how CPUs communicates with peripheral devices.
3. Apply C/Assembly programming techniques to simple problems.
4. Explain the idea behind memory hierarchy its use in memory caches and virtual memory.

Chapter 0
Introduction to Computing

Andrew Leung 5

5

Objective

- This chapter presents Background and History
  - What is a microprocessor?
  - What is the history of the development of the microprocessor?
  - How does transistor scaling affect processor design?
- This chapter reviews number system and basic logic circuits.
- This chapter reviews the basic operations of computer, especially how does the CPU retrieve and execute programs.
- If you do not familiar with the concept of a computer, you should pay more attention to read this chapter.

Andrew Leung 6

6

Objective

- This chapter presents Background and History
  - What is a microprocessor?
  - What is the history of the development of the microprocessor?
  - How does transistor scaling affect processor design?
- This chapter reviews number system and basic logic circuits.
- This chapter reviews the basic operations of computer, especially how does the CPU retrieve and execute programs.
- If you do not familiar with the concept of a computer, you should pay more attention to read this chapter.

Andrew Leung 7

7

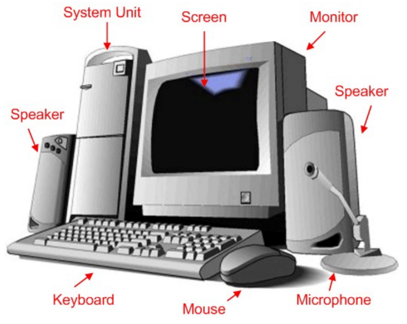
Section 0.0
Background and History

Andrew Leung 8

8

What is a Computer?

A **computer** is a programmable machine that receives input, stores and manipulates data//information, and provides output in a useful format.



9

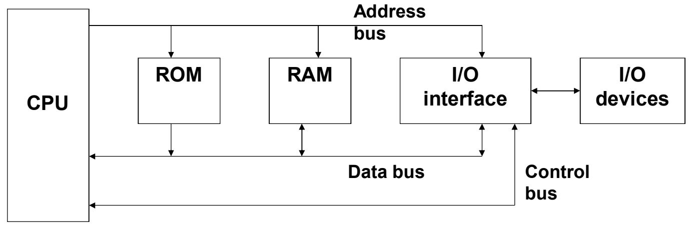
BLOCK DIAGRAM OF A BASIC COMPUTER SYSTEM

Basic computer system consist of a Central processing unit (CPU), memory (RAM and ROM), input/output (I/O) unit.

CPU: execute the program

Memory: store data and program

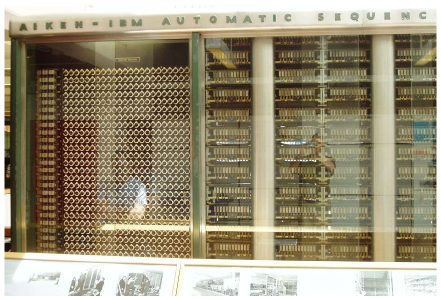
I/O: communicate with outside world



Block diagram of a basic computer system

10

In old days, CPU size is :



11

What is a Microprocessor

Microprocessor is a Central Processing Unit (CPU) on a single chip.

It contains millions of transistors connected by wires

Core i7 die
Picture: Intel

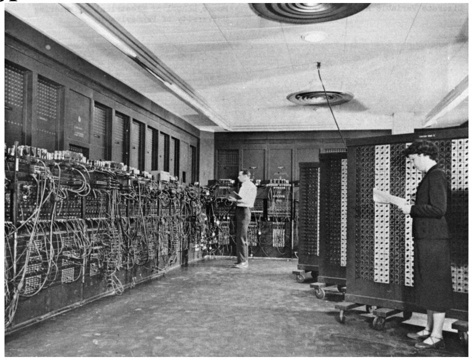
Core i7 in package
Picture: Ebbesen

12

Electrical Numerical Integrator and Calculator

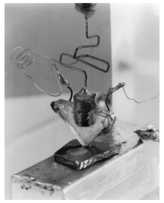
•**Designed for the U.S. Army's Ballistic Research Laboratory**
•**Built out of**
–**17,468 vacuum tubes (light bulb like transistors)**
–**7,200 crystal diodes**
–**1,500 relays**
–**70,000 resistors**
–**10,000 capacitors**
•**Consumed 150 kW of power**
•**Took up 72 squared meters**
•**Weighted 27 tons**
•**Suffered a failure on average every 6 hours**

Electrical Numerical Integrator and Calculator



(Picture: U.S. Army)

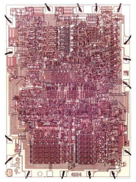
Transistors and Microprocessor



The First Transistor was Created in 1947. (Bell Labs)

The First Integrated Circuit (logic gates in a chip) was Created in 1959.

Intel Created the First Commercial Microprocessor
Introduced the 4004 in 1971, contained 2,300 transistors
Had roughly the same processing power as ENIAC

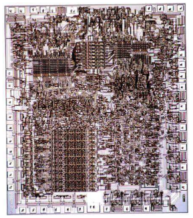


Intel 4004

IBM Introduced its Original PC in 1981

Used the Intel 8088 processor containing 29,000 transistors

Used operating system (MS-DOS) designed by Microsoft



Intel 8088

IBM PC Picture: Intel

Microprocessor Evolution

- •4004 transistors were 10 µm across
- •Pentium 4 transistors are 0.13 µm across
- •Human hair is about 100 µm across

•Smaller transistors allow

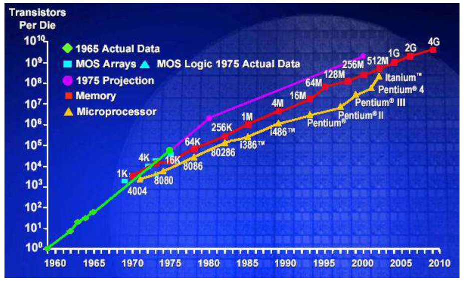
- –More transistors per chip
- –More processing per clock cycle
- –Faster clock rates
- –Smaller/cheaper chips

Andrew Leung 17

17

Moore’s Law

"The number of transistors incorporated in a chip will approximately double every 24 months.“ (1965)



Andrew Leung 18

18

Microprocessor Applications

NOT ONLY in conventional computers.

1. Mobile Phone
2. Mouse
3. Keyboard
4. Air Conditioner
5. Fax Machine
6. Fan
7. Toys

Andrew Leung 19

19

Section 0.1

Numbering and Coding Systems

(know how computers store data)

Andrew Leung 20

20

Outlines of Section 0.1

- Decimal and binary number systems
- Converting between decimal to binary
- Hexadecimal system
- Converting between binary and hex
- Counting in bases 10, 2, and 16
- Addition and subtraction of hex number
- ASCII codes
- *Please review the items if you have forgotten them.*

Number Representation

A positive number N can be written in positional notation as
$N = (a_{n-1}\, a_{n-2} \ldots a_1\, a_0)_r$
where $a_i \in (0,1, \ldots, r\text{-}1)$
r = radix or base of the number system being used
a_i = integer digit i when $n - 1 \geq i \geq 0$
a_{n-1}= most significant digit (MSD)
a_0 = least significant digit (LSD)

$$\textbf{decimal value} = a_{n-1}r^{n-1} + \ldots + a_1 r + a_0 r^0$$

Decimal number

There are ten digits,(0-9), in this system.
decimal number, r = 10

From the Polynomial from:
$(a_{n-1}\, a_{n-2} \ldots a_1\, a_0)_r \quad = a_{n-1}10^{n-1} + \ldots + a_0 10^0$

Example:
e.g. $(7392)_{10}$
$= 7\times10^3 + 3\times10^2 + 9\times10^1 + 2\times10^0$

Binary number

A binary number system has only two digits, called bits. A binary number is a weighted number. The right-most bit is the least significant bit (LSB) and the left-most bit is the most significant bit (MSB).

Example: non negative integer

Decimal Number	Binary Number	
	MSB	LSB
0	0	0
1	0	1
2	1	0
3	1	1

In general, for non negative integer, with n bits, we can count up from 0 to 2^n - 1.

Binary number

Binary-to-Decimal Conversion

From the Polynomial from:

$(a_{n-1}\, a_{n-2} \ldots a_1\, a_0)_r = a_{n-1}2^{n-1} + \ldots + a_0 2^0$

Example. Convert 0000 1101 to decimal form

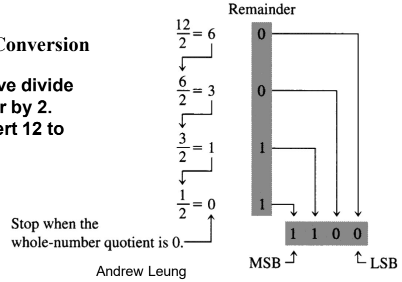
$0000\ 1101_2 = (1x2^3)+(1x2^2)+(1x2^0)$

$= 8 + 4 + 1 = 13$

0001 0101 = ?

Decimal-to-binary Conversion

Main step: Recursive divide the decimal number by 2. For example, convert 12 to binary number.



Hexadecimal Numbers

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Hexadecimal Numbers

Binary-to-Hexadecimal Conversion

Example. Convert 1100101001010111_2 to hexadecimal

Binary	1100	1010	0101	0111	
Hex.	C	A	5	7	= $CA57_{16}$

Hexadecimal-to-Decimal Conversion

- Convert the hexadecimal number to binary and then to convert from binary to decimal.
- Another way is to multiply the decimal value of each hexadecimal digit by its weight and then take the sum of these products.

Decimal-to-Hexadecimal Conversion

Repeated division-by-16 method.

Hexadecimal Numbers

Convert decimal number to HEXADECIMAL

1128 = ?

1128/16=70, remainder=8

70/16=4, remainder=6

4/16=0, remainder=4

==> 468_{16}

256=?

DIVISION	RESULT	REMAINDER (in HEX)
256 / 16	16	0
16 / 16	1	0
1 / 16	0	1
ANSWER		100

Hexadecimal Numbers

590=?

DIVISION	RESULT	REMAINDER (HEX)
590 / 16	36	E (14 decimal)
36 / 16	2	4 (4 decimal)
2 / 16	0	2 (2 decimal)
ANSWER		24E

29

Computer: Unsigned Binary Number

To represent non-negative number
Need to specify the number of bits used.

With n digits, 2^n unique numbers (from 0 to 2^n-1) can be represented.
If n=8, 256 (=2^8) numbers can be represented 0-255.

Convert 125 from decimal to 8 bit unsigned binary — Answer 0111 1101 (NOT 111 1101)

Convert 96 from decimal to 8 bit unsigned binary — Answer 0110 0000 (NOT 110 0000)

Convert 0001 0011 from binary to decimal — Answer 19

30

Computer: Unsigned Binary Number

Add the two 8-bit binary numbers (0011 1101) and (0001 0111).

$$\begin{array}{cccccccccc} & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ + & - & - & - & - & - & - & - & - \\ & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{array}$$

Subtract the two 8-bit binary numbers (0011 1101) and (0001 0111).

$$\begin{array}{ccccccccc} & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ - & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ & - & - & - & - & - & - & - & - \\ & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{array}$$

31

Computer: Unsigned Binary Number

Multiply two 8-bit binary numbers (0001 0111) and (0000 1010).

$$\begin{array}{ccccccccc} & & & & 1 & 0 & 1 & 1 & 1 \\ \times & & & & & 1 & 0 & 1 & 0 \\ \hline & & & & 0 & 0 & 0 & 0 & 0 \\ & & & 1 & 0 & 1 & 1 & 1 & \\ & & 0 & 0 & 0 & 0 & 0 & & \\ & 1 & 0 & 1 & 1 & 1 & & & \\ \hline & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{array}$$

32

Computer: Unsigned Binary Number

Division

```
                    1  1  0  1
1  0  0  1  |  1  1  1  0  1  1  1
               1  0  0  1
               -----------
                  1  0  1  1
                  1  0  0  1
                  -----------
                        1  0  1  1
                        1  0  0  1
                        -----------
          Remainder           1  0
```


Computer: Signed Binary Number

In mathematics, negative numbers in any base are represented in the usual way, by prefixing them with a "−" sign.

However, in computer hardware, numbers are represented in bit, so a method of encoding the minus sign is necessary. Modern computers typically use the two's-complement representation, but other representations are used in some circumstances

Computer: Signed Binary Number

1’s complement and 2’s complement

1's complement of a n-bit pattern is a transform that maps a n-bit pattern to another n-bit pattern. It simply changes all 1’s to 0’s and all 0’s to 1’s, as illustrated below:
Example, 1’s complement of a 8-bit pattern, 1011 0010=0100 1101

2's complement of a n-bit pattern is a transform that maps a n-bit pattern to another n-bit pattern
The 2's complement of a bit pattern is found by adding 1 to the LSB of the 1's complement.
2's complement = (1's complement) + 1 LSB bit

Example. Find the 2's complement of the binary number 010110010.

```
      0 1 0 1 1 0 0 1 0
      1 0 1 0 0 1 1 0 1   1's complement
+     0 0 0 0 0 0 0 0 1   Add 1 LSB bit
  ----------------------------
      1 0 1 0 0 1 1 1 0   2's complement
```


Computer: Signed Binary Number

2's complement representation

position number = same as the unsigned number,
negative number is the 2's complement of the corresponding positive number.

Example : express decimal numbers in the 8-bit 2's compl. representation

25 = 0001 1001
-25 = 1110 0111

32 = 0010 0000

-128 =1000 0000
-1 =1111 1111
12 =0000 1100
-12 =1111 0100

Computer: Signed Binary Number

From 2's complement to decimal:

If sign bit is 0, easy (same as "unsigned binary to decimal)
Example:
0000 11012 = $(1x2^3)+(1x2^2)+(1x2^0)$
= 8 + 4 + 1 = 13

If sign bit is 1, the magnitude of the negative number is equal to the decimal value of the 2's complement of the negative number.
Example
1110 0111=?
Sign bit =1 =>-ve
1110 0111 -> The magnitude 0001 1001 = 25=> -25

0011 0000=?
Sign bit =0=> +ve
48

Andrew Leung 37

37

Computer: Signed Binary Number

Arithmetic Operations with 2's Complement Number System

Addition or Subtraction can be reduced to Addition

Addition
There are four cases:
Case 1: Both numbers are positive.

```
    0000 0111      7
+   0000 0100      4
-------------------------
    0000 1011     11
```

Case 2: Positive number larger than negative number.

```
    00001111      15
+   11111010      -6
-------------------------  ---
  1 00001001       9
```

Discard carry
No need to use the carry bit to check overflow!
We use other method

Andrew Leung 38

38

Computer: Signed Binary Number

Arithmetic Operations with 2's Complement Number System

Case 3: Negative number larger than positive number.

```
    00010000      16
+   11101000     -24
-------------------------  ----
    11111000     - 8
```

The sum is negative and therefore is in 2's complement form.
Case 4: Both numbers are negative.

```
    11111011     - 5
+   11110111     - 9
--------------------------
  1 11110010     -14
```

Andrew Leung 39

39

Computer: Signed Binary Number

Arithmetic Operations with 2's Complement Number System

Overflow Condition

When two signed numbers are added and the number of bits required to represent the sum exceeds the number of bits in the two numbers, an overflow result is indicated by an incorrect sign bit.

An overflow can occur only when both numbers are positive or negative. 128+58

```
    01111101
+   00111010
----------------------------
    10110111
```

From the binary calculation, +ve + +ve = -ve => impossible => Overflow.

Check sign bits.

Andrew Leung 40

40

Computer: BCD code

Binary coded decimal (BCD) is a way to express each of the decimal digits with a binary code. Since there are only ten code groups in the BCD system, it is very easy to convert between decimal and BCD. Because we like to read and write in decimal, the BCD code provides an excellent interface to binary systems

8421 Code

BCD	decimal	BCD	decimal
0000	0	0101	5
0001	1	0110	6
0010	2	0111	7
0011	3	1000	8
0100	4	1001	9

16 = 0001 0110
15 = 0001 0101

Computer: BCD code

Rules to perform BCD addition:
Add in Binary but
(i) add 110 to the result if it is between 1010 and 1111
(ii) add 110 to the result if there is a carry

Example. Add the following two 2-digit BCD numbers:
00010110 + 00010101

```
 0001   0110
+0001   0101
-----------------------
 0010   1011   Right group is invalid (>9), left group
               is valid. Add 6 to invalid code
+
        0110
----------------------
 0011   0001          31
```


Computer: ASCII

ASCII
The most widely used character code in computer applications is the *ASCII* (American Standard Code for Information Interchange) code. Example. Encode the word Boy in ASCII code, representing each character by two hexadecimal digits.

Character	Binary Code	Hexadecimal Code
B	0100 0010	42
O	0110 1111	6F
Y	0111 1001	79

Computer: Examples

Example.
Determine the decimal value of the number 11101000.

Ans: ???? (We do not know because we do not know the format)

Now, if we know the format :
if it is a binary number of
1. 8-bit unsigned system representation
2. 8-bit 2's complement system representation
3. BCD representation

1) 232
2) -24
3) invalid because 1110 = ? In BCD

Computer: Examples

Example.
Determine the decimal value of the number 0100 0001.

if it is a binary number of
1. 8-bit unsigned system representation
2. 8-bit 2's complement system representation
3. BCD representation

1) 65
2) 65
3) 41

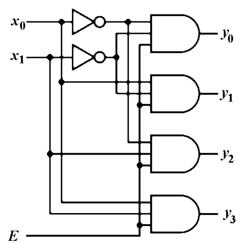
Section 0.2
Digital Primer

Outlines of Section 0.2

- Binary Logic
- Logic Gates - AND, OR, NOT, XOR, NAND, NOR
- Logic design using gates
- Decoders
- Flip-flops
- *Please review the items if you have forgotten them.*

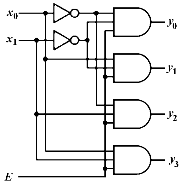
Decoder

A decoder can take the form of a multiple-input, multiple-output logic circuit that converts coded inputs into coded outputs, where the input and output codes are different. e.g. n-to-2^n, binary-coded decimal decoders.



Decoder

A decoder can take the form of a multiple-input, multiple-output logic circuit that converts coded inputs into coded outputs, where the input and output codes are different. e.g. n-to-2^n, binary-coded decimal decoders.



E	X1X0	Y3	Y2	Y1	Y0
0	XX	0	0	0	0
1	00	0	0	0	1
1	01	0	0	1	0
1	10	0	1	0	0
1	11	1	0	0	0

Decoding is necessary in applications such as data multiplexing, 7 segment display and memory address decoding.

Section 0.3
Inside the Computer

Important Terminology

- Bit 0
- Nibble 0000 (4 bits)
- Byte 0000 0000 (8 bits)
- Word 0000 0000 0000 0000 0000 0000 0000 0000 (32 bits)
- KB (kilobyte) = 2^{10} bytes
- MB (megabyte) = 2^{20} bytes
- GB (gigabyte) = 2^{30} bytes
- TB (terabyte) = 2^{40} bytes

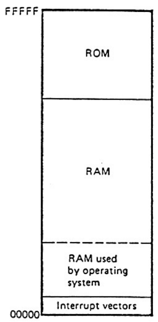
ROM v.s. RAM

- ROM (read only memory)
  - ROM contains programs and information essential to operation of the computer.
  - for permanent data which cannot changed by the user
  - called as nonvolatile memory
  - Data does not lost when power off
- RAM (random access memory)
  - for temporary storage of programs that it is running
  - Data lost when power off
  - called as volatile memory

Memory

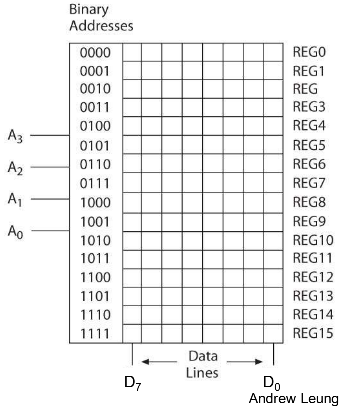
The memory is used to store program and data. The memory is arranged sequentially into a number of units, each capable of holding one computer word. Each unit is referred to as a memory location and identified by a unique address. The binary word stored in a particular location is referred to as the content of that location.

Binary Address	Memory Content 8 bit content
	:
:	:
:	:
0100	:
0011	content of address 0011
0010	content of address 0010
0001	content of address 0001
0000	content of address 0000



53

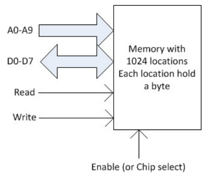
Memory



- A semiconductor storage device consisting of registers that store binary bits
- Two major categories
  - Read/Write Memory (R/W)
  - Read-only-Memory (ROM)

54

Memory



Address lines: Wires carry an address of a location being accessed. (unidirectional from external circuit from memory)

Data lines: Wires carry the data content being transferred between external circuit and memory (bidirectional)

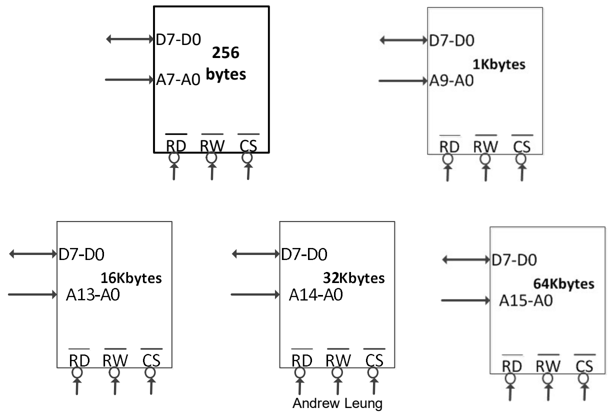
Enable: a wire carries to a control signal to enable the memory chip. (unidirectional from external circuit from memory)

Read: a wire carries to a control signal to instruct the memory chip the operation is a read operation. (unidirectional from external circuit from memory)

Write: a wire carries to a control signal to instruct the memory chip the operation is a write operation. (unidirectional from external circuit from memory)

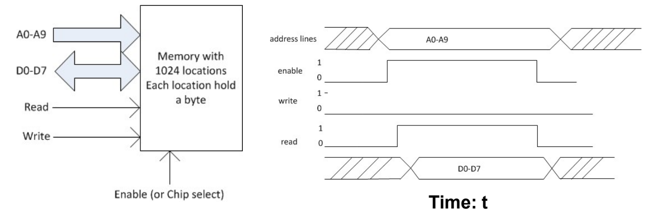
55

Memory



56

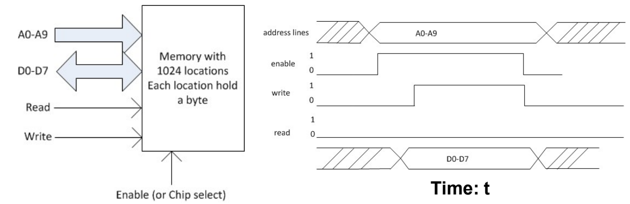
Memory



Read a byte from a location

- External circuit sets the address of the location to address lines
- External circuit enables the chip enable
- External circuit enables Read
- The memory chip then puts the content of the selected location on D0-D7.
- Now extern circuit can access the data from D0-D7

Memory



Write a byte to a location

- External circuit sets the address of the location
- External circuit enables the chip
- External circuit puts the data on D0-D7
- External circuit enables the Write line
- The memory chip then saves incoming data on the corresponding location.

Basic Computer Architecture

Computer:

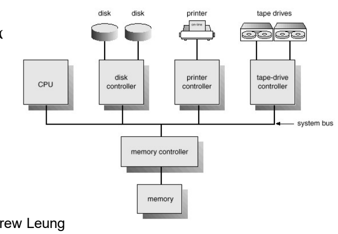
Perform a number of elementary computer operations (**instructions**) that manipulate information, called **data**.

Instruction Set:

The collection of all the instructions is called the instruction set of that computer.

Program: Consists of a number Instructions.

A digital computer may be divided into
The Central Processing Unit (CPU),
The Memory,
The Input/Output Devices (I/O).



Basic Computer Architecture

Computer:

Execute the programs by run the instructions one-by-one.

1. Load an instruction from memory to CPU.
2. Execute the instruction.
3. During the execution, the CPU may need to read or write the data from or to the memory.

Questions: How to access an instruction or data item from the memory?
How to make sure that the instructions are executed in a correct order?

Basic Computer Architecture

Input units: Computer accepts information through input units, which read the data. (keyboard, mouse). Each input unit has some corresponding addresses.

Memory: Store programs and data.

A program contain many instructions stored in memory.

Each memory unit or location has **an address**.

- Primary storage and secondary storage
- *Primary* (IC chips in the computer) – to hold data and programs which are currently used.
- *Secondary* (disk) – to hold large amount of data or programs which are not currently used.
- In some computer systems, such as 6502, program and data share the same memory space.
- In some computer systems, such as 8051, program and data are with separate memory spaces.

Output units: Counterpart of the input unit. Send processed results to the outside world. (monitor). Each output unit has some corresponding addresses.

Basic Computer Architecture

CPU: **Arithmetic and Logic Unit** (ALU), **and Control Unit**

Arithmetic logic unit (ALU). The ALU performs all the numerical computations and logical evaluations for the processor. The ALU receives data from the memory, performs the operations, and, if necessary, writes the result back to the memory.

Control Unit: The control unit contains the hardware instruction logic. The control unit decodes and monitors the execution of instructions. The control unit also acts as an arbiter as various portions of the computer system compete for the resources of the CPU. Coordinate the operations of I/O units, memory, and ALU. It is effectively the nerve center that sends control signals to other units and senses their states.

Remark: In order to access a data item or an instruction, CPU needs to know their addresses. How ?

Basic Computer Architecture

In modern computer systems,

- CPU, memory, and a number of device controllers are connected to the system bus.
- I/O devices and the CPU can execute concurrently.
- Each device controller is in charge of a particular device type.
- Each device controller has a local buffer (a small number of regristers).
- CPU moves data from/to main memory to/from the local buffers.
- I/O is from the device to local buffer of controller.
- Device controller can inform CPU that it has finished its operation. (by using interrupt)

Bus

- CPU is connected to memory and I/O through strips of wire called a bus.
- Buses are used to Communicate between the computer components.
  - Data Bus
  - Address Bus
  - Control Bus

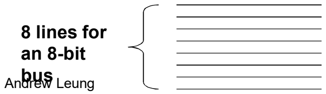
The Operation of Bus (see the figure in P.44)

- The CPU puts the address on the address bus, and the decoding circuitry finds the device.
- The CPU uses the data bus either to get data form that device or to send data to it.
- The control buses are used to provide read or write signals.
- The address bus and data bus determine the capacity of a given CPU.

65

Data Bus (see the figure in P.48)

- Size of data buses is larger, the better the CPU is.
  - Example：8 bits（slow）, 16 bits, 32 bits, 64 bits（fast）.
  - An 8-bit data bus can send 1 byte a time.
- Data buses are bi-directional.
- More data buses mean a more expensive CPU and computer.
- The processing power of a computer is related to the size of its buses.



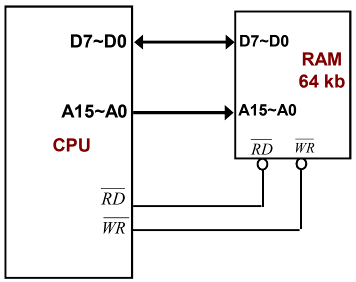
66

Address Bus (see the figure in P.48)

- Size of address buses is larger, the larger the number of devices and memory locations that can be addresses.
  - Example：8 bits（small）, 16 bits, 32 bits, 64 bits（large）.
  - A 16-bit address bus can indicate 2^{16}=64K bytes of addressable memory locations.
  - Regardless of the size of the data bus.
- Address buses are unidirectional.
- The number of address lines determines the number of locations with which a CPU can communicate.

67

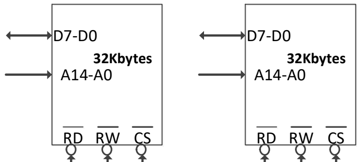
Single Structure (memory)



68

Simple memory structure

- CPU has 64 KB memory
- Two memory chips. Each is 32 KB
- Two RAM area is from 0000h to 7FFFh (RAM0) and 8000h to FFFFh (RAM1)



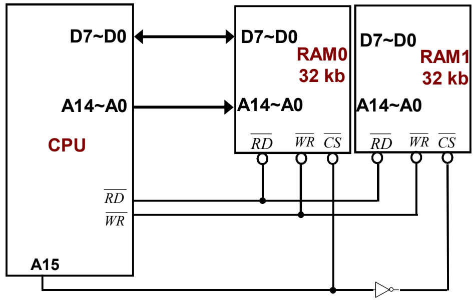
	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RAM0 0000 to 7FFF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RAM1 1000 to FFFF	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Andrew Leung 69

69

Simple memory structure

- There is two 32 kb RAM
- A15 applied to select one RAM chip
- Two RAM area is from 0000h to 7FFFh (RAM0) and 8000h to FFFFh (RAM1)



Andrew Leung 70

70

Address Decoding

- Given an n-address, build a circuit to detect a range of addresses
- Example: Given a 16-bit address (A15-A0), design a design to detect 0000h to 7FFFh (**assume active low**. When the address is in the range, the output of decoding circuit is 0.)

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1000 to 7FFF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

=> **Use A15**

Andrew Leung 71

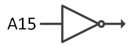
71

Address Decoding

- Example: Given a 16-bit address (A15-A0), design a design to detect 1000h to FFFFh (**assume active low**. When the address is in the range, the output of decoding circuit is 0.)

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1000 to FFFF	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

=> **Use $\overline{A15}$**



Andrew Leung 72

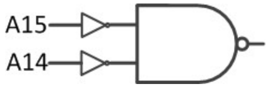
72

Address Decoding

❑Example: Given a 16-bit address (A15-A0), design a design to detect 0000h to 3FFFh (**assume active low**. When the address is in the range, the output of decoding circuit is 0.)

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0000-3FFF	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x

=> **Use $\overline{A15}$ and $\overline{A14}$**

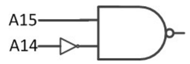


Address Decoding

❑Example: Given a 16-bit address (A15-A0), design a design to detect 8000h to FFFFh (**assume active low**. When the address is in the range, the output of decoding circuit is 0.)

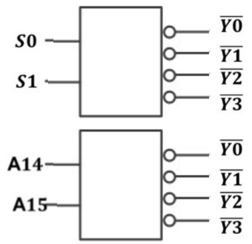
	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
8000-BFFF	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x

=> **Use A15 and $\overline{A14}$**



Address Decoding

❑ Use decoder

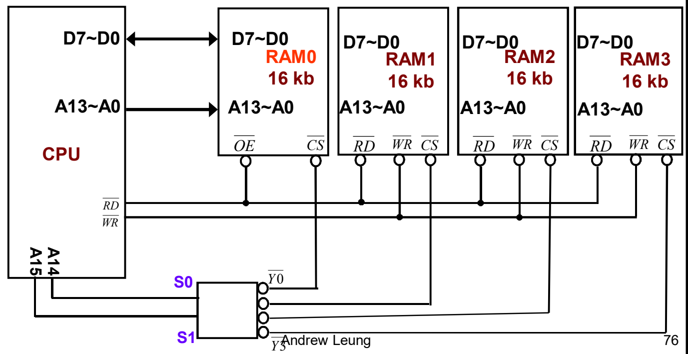


S1	S0	$\overline{Y0}$	$\overline{Y1}$	$\overline{Y2}$	$\overline{Y3}$
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

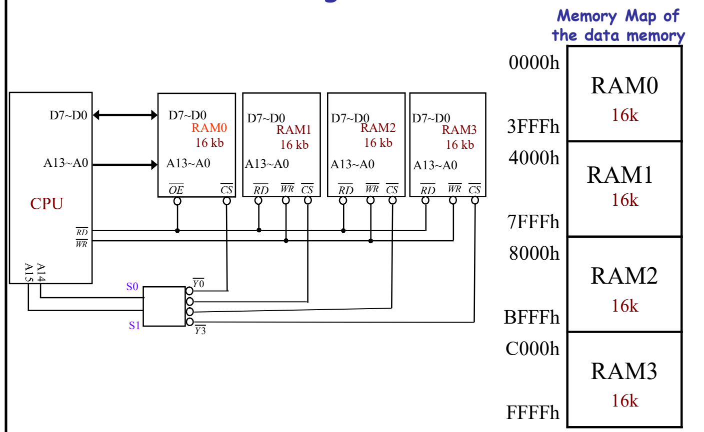
$\overline{Y0}$ detects A15 A14 = 0 0, Address range 0000H---3FFFF
$\overline{Y1}$ detects A15 A14 = 0 1, Address range ??
$\overline{Y2}$ detects A15 A14 = 1 0, Address range ??
$\overline{Y3}$ detects A15 A14 = 1 1, Address range ??

Single Structure

❑ There is four 16 kb RAMs
❑ Use A14 and A15 to select one RAM chip



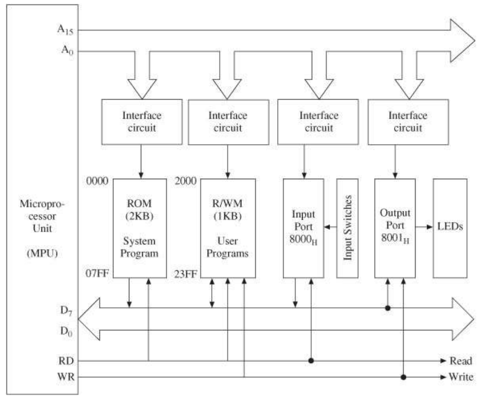
Single Structure



Andrew Leung 77

77

Example Microprocessor System



Andrew Leung 78

78

Operation

The operation of a computer can be summarized as follows:

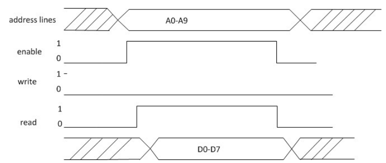
- The computer accepts information in the forms of programs and data through an input unit and stores it in the memory.
- Fetch the instructions one-by-one into the CPU.
- Decode and then perform the instructions.
- Information stored in the memory is fetched, under program control, into ALU, where it is processed.
- Processed information leaves the computer through output units.
- All activities inside the machine are directed by the control unit.

Andrew Leung 79

79

Example CPU Read a byte from memory (read instructions or data item)

- CPU puts the address of the location on address bus
- The decoding circuit selects the chip
- CPU sets the Read
- The memory chip then puts the content of the selected location on the data bus..
- CPU now can access the data.
- And then, … …

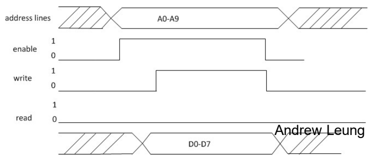


Andrew Leung 80

80

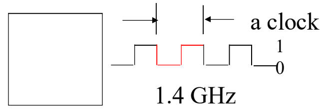
Example CPU Write a byte to memory (write data item)

- CPU puts the address of the location on address bus
- The decoding circuit (set the enable pin) selects the chip.
- CPU puts the data item one data bus.
- CPU sets the write signal.
- The memory chip then takes the data from the data bus and then saves it to the selected location (indicate by the address bus.
- And then …, …,



Machine Clock

- CPU performs the operations in the step-by-step manner.
- A CPU can be considered as a synchronic sequence logic circuit.
- Clock is used to synchronize work of the components on the machine.
- Clock decides the performance of the computer.



Central Processing Unit (CPU)

Inside a CPU, there are

- Arithmetic/logic unit (ALU): to perform the arithmetic and logic operation
- Control unit : coordinating the machine's activities
- Registers : to store temporary data

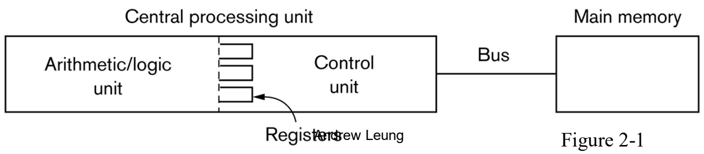
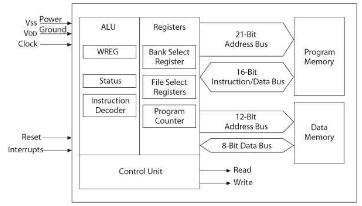


Figure 2-1

Inside CPU

- Accumulator and General Purpose Registers
  - Accumulator (WREG)
  - Registers
- Special-purpose registers
  - Program Counter (PC)
  - Instruction Register (IR)
  - Stack Pointer
  - Status Word Register
  - In PIC18, there are Bank and File Select Registers
    bank select register: select which data space in RAM is active
    file select register: for indirect addressing.



- Stack Point permits "context switching" in interrupt service routines (ISR) and subroutines

Inside CPU

Accumulator (WREG)

A register stores the results of arithmetic and logic operation.

Data Registers and Address Registers.

Registers stores data or addresses.

Program Counter (PC)

Store the address of the instruction to be executed in the next instruction cycles.

Is updated automatically.

Instruction Register (IR)

Store the instruction currently being executed or decoded

Inside CPU

Stack Pointer

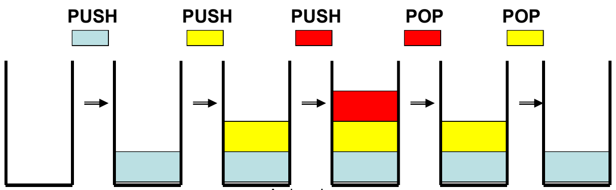
- Help the implementation of a stack data structure.

Status Word

- Contain several bits.
- Each bit indicates the states of CPU.
- For example, a Carry flag indicates if the last addition operation produces a carry or not.

Stack

- Stack：a section of RAM to store data items
- Stack register (stack pointer): point to the location of the top of the stack.
- Two operations on the stack：
  - PUSH：put an item onto the *top* of the stack
  - POP：remove an item from the *top* of the stack



Software: From Machine to High-Level Languages

- Machine Language: binary instructions
  - Difficult to decipher and write
  - Prone to cause many errors in writing
  - All programs converted into the machine language of a processor for execution

Instruction	Hex	Mnemonic	Description	Processor
10000000	80	ADD B	Add reg B to Acc	Intel 8085
00101000	28	ADD A, R0	Add Reg R0 to Acc	Intel 8051
00011011	1B	ABA	Add Acc A and B	Motorola 6811

Software: From Machine to High-Level Languages

- Assembly Language: machine instructions represented in mnemonics
  - Has one-to-one correspondence with machine instructions
  - Efficient in execution and use of memory; machine-specific and not easy to troubleshoot

Software: From Machine to High-Level Languages

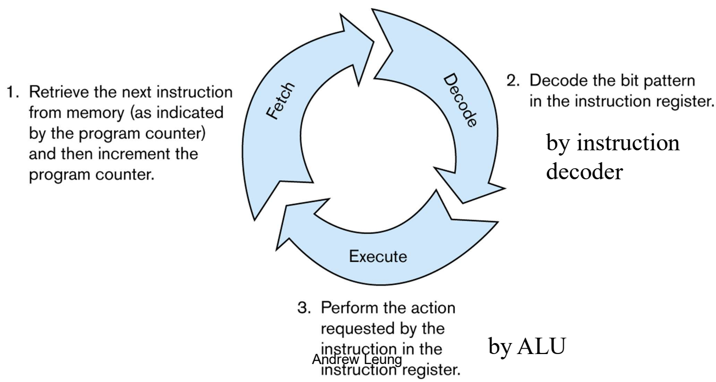
- High-Level Languages (such as BASIC, C, and C++)
  - Written in statements of spoken languages (such as English)
    - machine independent
    - easy to write and troubleshoot
    - requires large memory and less efficient in execution

The Machine Cycle（1/2）

- Every instruction in memory is executed by three steps:

  Fetch → Decode→ Execute
  - Each instruction has its micro-instruction (or micro-operations).
    - A micro-operation is an elementary operation that can be performed in parallel during one clock pulse period.
  - CPU has separate inside units for performing fetch/decode/execution.
  - The instruction decoder is to interpret the instruction fetched into the CPU.

The Machine Cycle（2/2）



Internal Organization of Computers

Example

- 8-bit data bus
- 16-bit address (for a total of 10000H locations)
  - address 0000-FFFFH

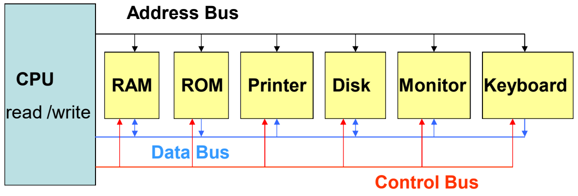


Figure 0-10 Internal Organization of Computers

Program Example 0-1

Action	Machine Code
Move value 21H into register W	0E21
Add value 42H to register W	0F42
Add value 12H to register W	0F12
No operation	0000
GOTO 0000	EF00 F000

```
Main: ORG        0x0000
      MOVLW      0x21
      ADDLW      0x42
      ADDLW      0x12
      NOP
      GOTO       Main
      END
```


Inside Memory

Memory address	Contents	Meaning
0000	0E21	instruction for loading a value to W register
0002	0F42	instruction for adding a value into W register
0004	0F12	instruction for adding a value into W register
0006	0000	No operation
0008	EF00	GO Back 0000
	F000	

Actions Performed by the CPU (1/2) (each step spend to a number of clock cycles)

1. The **PC** is reset to the value 0000H, indicating the address of the first instruction code to be execution. PC=0000
2. The CPU puts 0000H on the address bus and sent it out. The **PC** is added by 2. PC=0002
3. The memory circuitry finds 0000H while the CPU activates the READ signal, indicating to memory that CPU wants the two byte at location 0000H.
4. The content of memory locations 0000H and 0001H, which is 0E21, is put on the data bus and brought into the CPU. PC=0002
5. The CPU decodes the instruction 0E21

Actions Performed by the CPU（2/2）

6. The CPU know that it is a move instruction and the data is in the second part of the instruction. The CPU performs the moving instruction.

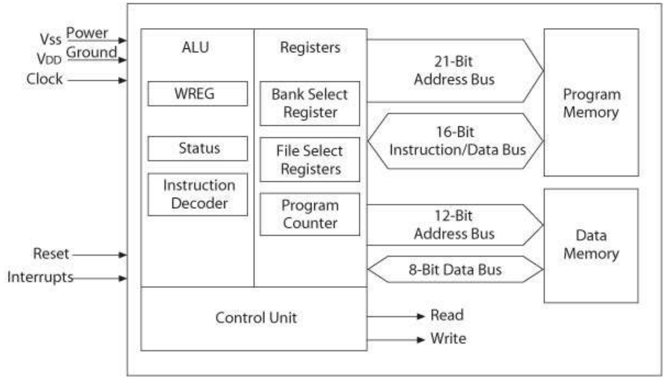
PC=0002
W=21

7. The CPU fetch the memory location 0002H.
The CPU *fetches* instruction 0F42. The PC add 2.
PC=0004

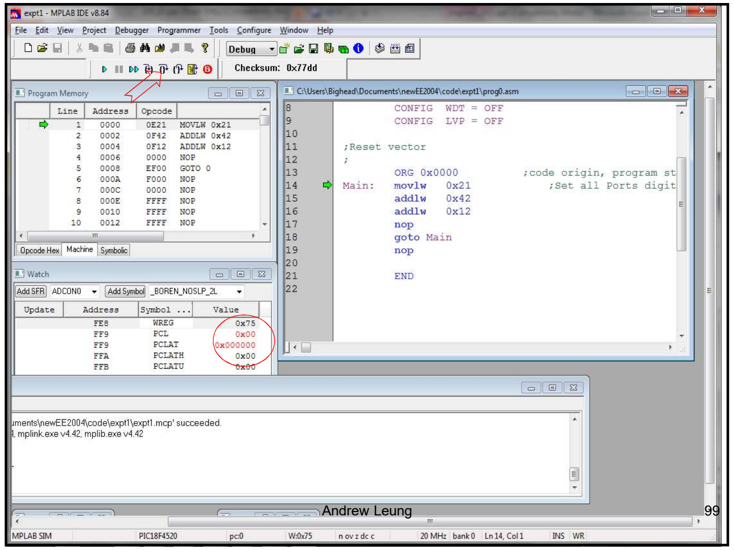
The CPU *decodes* 0F42. It is an ADD instruction. The data 42 is in the second part of the instruction. The destination is W register.
The ALU *executes* the add instruction and sets the result (63H) to register W.
PC=0004
W=63

97

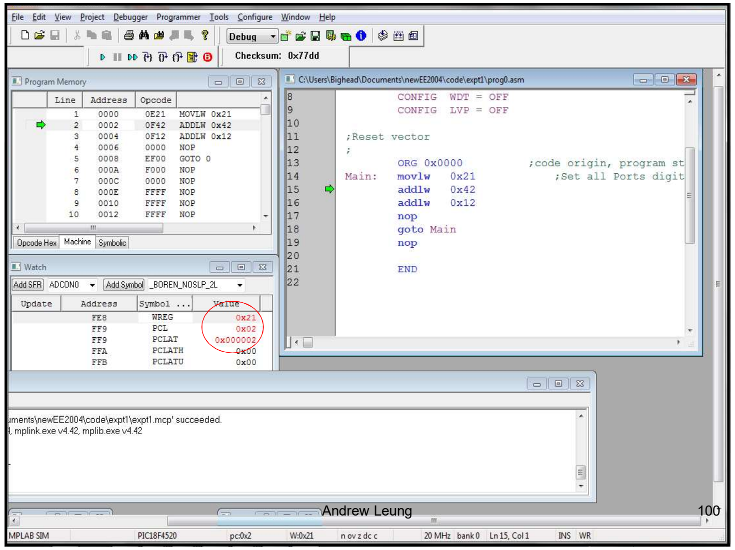
PIC18F4520



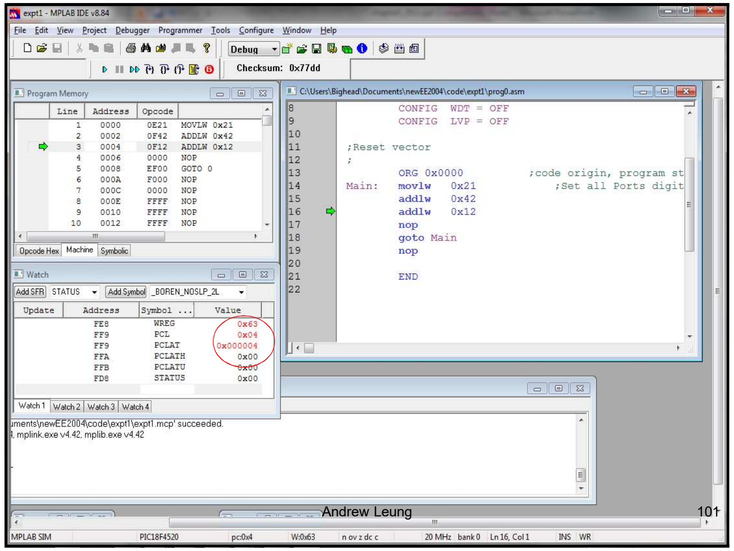
98



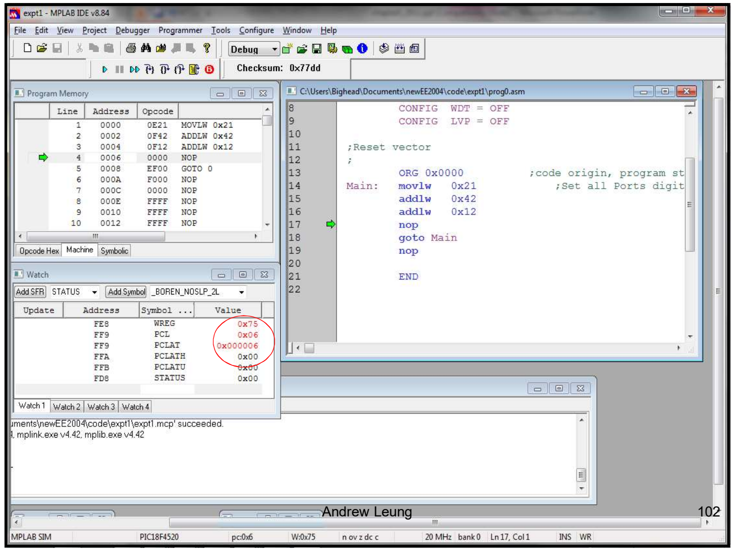
99



100



101



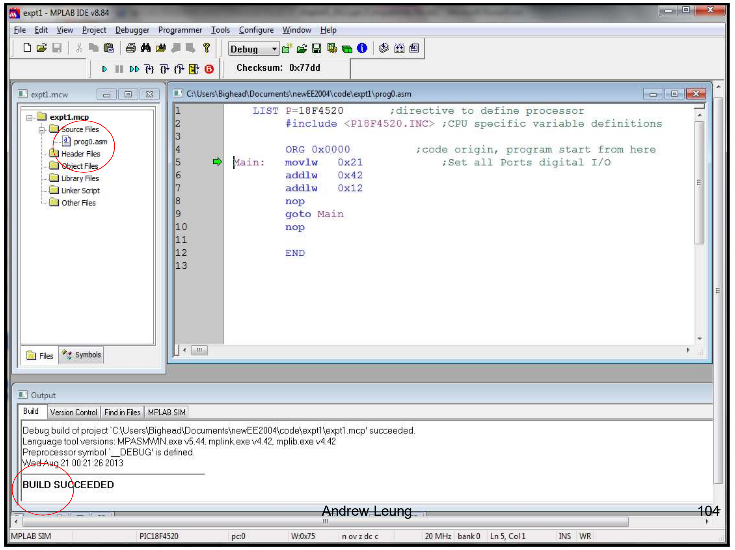
102

Example Procedure

1. Execute the MAPLAB IDE program.
2. Click “File”; “New” and type the code into the file editor window.
3. Click “File”, “Save As”. Create and Select the “My Documnet\Code\Chapter0” folder and type “prog0.asm’ as the program file name. (Make sure you save the file into the Chapter0 folder).
4. Click “Project”, “Project Wizard…”, “Next >”, select device “PIC18F4520”, click “Next >”, select “Microchip MPASM Toolsuite”, click “Next >”
5. Browse into folder “My Documnet\Code\Chapter0” , type “Exp0” as the Project file name and click “Save”.
6. Click “Next >”, expand the folder tree and locate the file prog0.asm. Click “Add >>” and “Next >” to put the prog0.asm file to the Project. Check the project parameters list and click “Finish” to finish the project definition process.

Andrew Leung 103

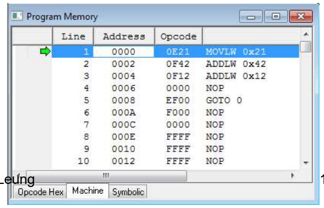
103



104

Example Procedure

7. Click "Project", "Build All" and select "Absolute".
8. "*BUILD SUCCESSED*" should appear at Output window. Should "BUILD FAILED" appear instead, check for the error messages, fix any errors found and repeat the build process 7 until success.
9. Click "File", "Save Workspace" to save your work. It will save all your current project related parameters into the file with extension "mcw". You can double click file Expt1.mcw later to continue your development.
10. Click "Debugger", "Select Tool", "4 MPLAB SIM" to select MPSIM simulator. Screen look remain unchanged, however. MPSIM takes the control.
11. Click "View", "Program Memory" (you see the content of program memory)

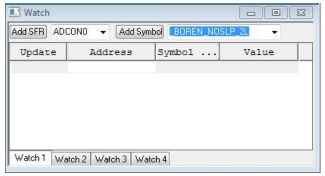


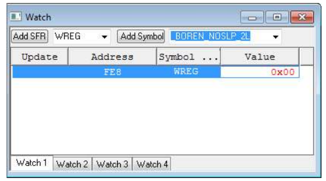
Andrew Leung 105

105

Example Procedure

12. Click "View", "Watch"
    Now you can see the content of some registers.
    Select "WREG" and then click "add SFR"
    Now you can watch W register during the simulation.



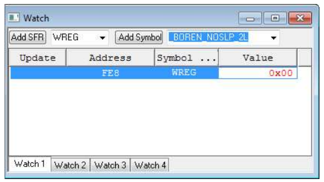


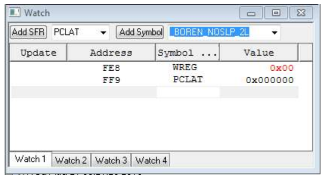
Andrew Leung 106

106

Example Procedure

Select "PCLAT" and then click "add SFR"
Now you can watch PC counter during the simulation.



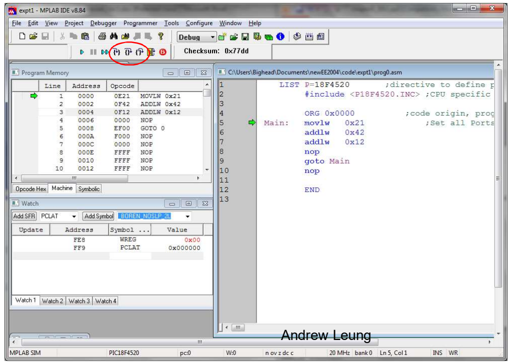


Andrew Leung 107

107

Example Procedure

12. Click "Step Over".
    We can observe how contents of register changes during the execution of the program.



Andrew Leung 108

108