Chapter 1

Microprocessor architecture

ECE 3120

Dr. Mohamed Mahmoud

http://iweb.tntech.edu/mmahmoud/
mmahmoud@tntech.edu
Outline

1.1 Computer hardware organization
   1.1.1 Number System
   1.1.2 Computer hardware organization

1.2 The processor

1.3 Memory system operation

1.4 Program Execution

1.5 HCS12 Microcontroller
1.1.1 Number System

- Computer hardware uses binary numbers to perform all operations.
- Human beings are used to decimal number system.
- Conversion is needed to convert numbers between the internal (binary) and external (decimal) representations.
- Octal and hexadecimal numbers have shorter representations than the binary system.
- The binary number system has two digits 0 and 1
- The octal number system uses eight digits 0 and 7
- The hexadecimal number system uses 16 digits: 0, 1, .., 9, A, B, C,.., F
- A prefix is used to indicate the base of a number.

<table>
<thead>
<tr>
<th>Base</th>
<th>Prefix</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>%</td>
<td>%10001010</td>
</tr>
<tr>
<td>Octal</td>
<td>@</td>
<td>@123467</td>
</tr>
<tr>
<td>Decimal</td>
<td></td>
<td>12345678</td>
</tr>
<tr>
<td>Hexadecimal (shorthand hex)</td>
<td>$</td>
<td>$392</td>
</tr>
</tbody>
</table>

- Convert %1000101 to Hexadecimal

0100 0101 = $45

- Convert $4F to Binary

$4F = 0100 1111

- Unsigned numbers are always positive

- Signed number

B010 0000

- Most significant bit (B) = 1 → negative, otherwise positive
1- **Unsigned number**

\[ \%1111 = 1 + 2 + 4 + 8 = 15 \]

\[ \%0111 = 1 + 2 + 4 = 7 \]

Unsigned N-bit number can have numbers from 0 to \(2^{N-1}\)

2- **Signed number**

\%1111 is a negative number.

To convert to decimal, calculate the two’s complement

The two’s complement = one’s complement +1 = \%0000 + 1

\%=\%0001 = 1 \rightarrow \text{then} \%1111 = -1

\%0111 is a positive number = 1 + 2 + 4 = 7.

- Unsigned 8-bit number can have numbers from 0 to 255
  \%00000000 to \%11111111

- Signed 8-bit number can have numbers from -128 to +127
  to \%10000000 to \%01111111
- Signed N-bit number can have numbers from $-2^{N-1}$ to $2^{N-1}-1$
- Unsigned 3-bit number can represent numbers from 0 (= %000) to 7 (=111)
- Signed 3-bit number can represent numbers from -4 (= %100) to 3 (= %011)

Computer does not know whether a bit sequence represents a signed or an unsigned number -> **It is the programmer’s responsibility!**
1.1.2 Computer hardware organization

- A Computer consists of **hardware** and **software**
- The hardware consists of **processor, input devices, output devices** and **memory**.
1- Processor

- Also called Microprocessor (MP) or Central Processing Unit (CPU).
- Executes programs.
- Performs: (1) all of the computational operations and (2) the coordination of the usage of resources of a computer.
- Ex. can a memory and an input device write data on a shared bus simultaneously? NO, control signals are issued by processor to coordinate the devices operations.

2- Input devices

- Used to enter the programs to be executed and the data to be processed into the computer.
- Examples: keyboard, keypad, switches, sensors, scanners, bar code readers, etc.
3- Output devices
- The results of the operations done by the processor should be displayed or printed on a media so that the user can see them.
- Examples: seven segment displays, light emitting diode (LED), liquid crystal displays (LCD), a monitor, etc.

4- Memory
- Programs to be executed and data to be processed are stored in memory so that the processor can readily access them.

5- Buses
- Address bus: The set of conductor wires that carry address signals.
- Data bus: The set of conductor wires that carry data signals.
- Control bus: The set of conductor wires that carry control signals.
Outline

1.1 Computer hardware organization

1.2 The processor

1.3 Memory system operation

1.4 Program Execution

1.5 HCS12 Microcontroller
- A processor consists of **arithmetic logic unit (ALU)**, **control unit**, and **registers**.

- The ALU performs the arithmetic and logic operations requested by the program.

- A simple ALU that can perform only four operations (addition, summation, AND, OR) is shown in Figure 1.1.

![ALU Diagram](image)

*Figure 1.1 An ALU that implements ADD, SUB, AND, and OR operations*
- All four operations are performed simultaneously by different circuits whereas the instruction opcode tells the multiplexer to select one of the four units’ outputs as the result.

- The adder is used to perform addition and subtraction operations.

- The ALU is more complicated if the processor designer wants to implement more operations directly in the hardware.

Remember how multiplexer works:

![Multiplexer Diagram]

<table>
<thead>
<tr>
<th>Select</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>I₀</td>
</tr>
<tr>
<td>01</td>
<td>I₁</td>
</tr>
<tr>
<td>10</td>
<td>I₂</td>
</tr>
<tr>
<td>11</td>
<td>I₃</td>
</tr>
</tbody>
</table>
The ALU operations:-

1- **OpCode = 00**

- The comparator output = 0, MUX2 output = CIN, MUX1 output = B, the adder inputs are n-bit A and n-bit B.
- The result of the adder is $A + B + CIN$, CIN is the input carry.
- MUX3 selects the output of the adder and the carry.
2- **Opcode = 01**

- The comparator output = 1, MUX2 output = 1, MUX1 output = B' (one’ complement of B), the adder inputs are n-bit A and n-bit B’.
- The result of the adder is A + B’ + 1 = A - B. Remember B’ + 1 = - B. B’ + 1 is the two’s complement of B.
- MUX3 selects the output of the adder and the carry (called borrow).
3- Opcode = 10

MUX3 selects the output of the AND. It is the n bits resulted from performing logic AND operation for A and B.

4- Opcode = 11

MUX3 selects the output of the OR. It is the n bits resulted from performing logic OR operation for A and B.

Remember:-

X can be zero or one

The ALU in Fig. 1.1 can be expanded to perform more instructions
2- Control Unit

- The control unit decodes **machine codes** and performs the operations specified by them.

- A machine instruction has a mandated field called the **opcode** which can tell what operations are needed to execute the instruction. Other fields are optional and used to specify the operands to be operated on.

- A machine instruction is a combination of 0s and 1s.

- Instruction execution is timed by a **clock signal**. A clock signal is a square signal. At each cycle, some operations are performed.

- A program usually has multiple instructions that are normally stored in sequential locations in memory.
- A **program counter (PC)** is used to keep track of the address of the instruction to be executed next.

- The first step to execute an instruction is to fetch it from the memory location pointed by PC and place it in the **instruction register (IR)** where it is decoded, and executed.

- The result of the decoding process is appropriate control signals to execute the instruction.

- Whenever the processor fetches an instruction, the PC will be incremented by the length of that instruction so that it points to the **next instruction**.

- Both PC and IR are two registers inside the microprocessor.

- The processor may not execute instructions in sequence due to the need to execute instructions based on the condition or the need to repeat a certain group of instructions.

- The processor uses **conditional** and **unconditional branch** (or **jump**) instructions to change the program flow.
- Conditional branches are performed when a condition is satisfied, e.g., the previous instruction caused a carry.

- After executing an instruction, the **condition code register (CCR)** is usually automatically updated to reflect the result of the instruction.

- The CCR is usually used to make a condition in conditional branch. Each bit in the register is independent and has a different indication.

- Example, a zero flag bit is set if the instruction result is zero, a sign flag bit is 1 if the result is negative and so on.

- **EX.:** \( A = A - B \)

  If zero flag = 1 jump to --- jump if \( A = B \)
3- Registers

- A register is a storage location inside the CPU.
- A register is used to hold data or address during the execution of an instruction.
- A register, very close to the ALU, provides fast access to operands for program execution.
- The number of registers varies from processor to processor.
- Unlike memory, registers are not used to store the program or large data.
- Accessing memory is much slower than accessing registers.
2- Computer software

- Computer programs are known as software

- A program is a sequence of instructions

- **Machine instruction:** A sequence of binary digits which can be executed by the processor

  0001 1000 0000 0110:  A ← [A] + [B]
  0100 0011:  A ← [A] + 1

- **Machine instruction:** Hard to understand, enter, debug, and maintain for human being

- **Assembly language:** An assembly instruction is a mnemonic representation of a machine instruction.

  Examples:  ABA  ; A ← [A] + [B]
             DECA  ; A ← [A] − 1
- **Assembler**: A program that translates an assembly language program to the equivalent machine code that can be executed by the microprocessor.

- **Compiler**: A program that translates a high level programming language to the equivalent machine code.

- **Source code**: A program written in assembly or high-level language

<table>
<thead>
<tr>
<th>line</th>
<th>Address</th>
<th>Machine code</th>
<th>Source Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>org $2000</td>
</tr>
<tr>
<td>2</td>
<td>2000</td>
<td>B6 10 00</td>
<td>ldaa $1000</td>
</tr>
<tr>
<td>3</td>
<td>2003</td>
<td>BB 10 01</td>
<td>adda $1001</td>
</tr>
<tr>
<td>4</td>
<td>2006</td>
<td>BB 10 02</td>
<td>adda $1002</td>
</tr>
<tr>
<td>5</td>
<td>2009</td>
<td>7A 11 00</td>
<td>staa $1100</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>end</td>
</tr>
</tbody>
</table>
Compiler vs assembler

- Compiler is more complex than assembler

- Each assembly language instruction is translated to one machine language instruction

- But, one statement written in high level language may be translated into tens or hundreds of machine language instructions and this translation is not usually optimal

- Assembly language uses mnemonic symbol to represent each machine translation

- Using symbols can make programming much easier than using binary machine codes

- The microprocessor can execute only machine language instructions
Microprocessor VS Microcontroller

- A microprocessor (MP) needs external devices such as memory, I/O ports, etc to run a program.

- A microcontroller (MCU) contains a microprocessor, memory and I/O ports within a single chip.

- It may also contain timer, serial communication interface, and A/D (analog/digital) and D/A converters.

- Microcontrollers have been used in almost every product that requires a certain amount of intelligence.

- Microcontrollers have been used in printers, modems, MP3 players, cars, home appliances such as washing machines, microwave ovens, etc.
Embedded Systems

- An embedded system is a **special purpose computer system** designed to perform a **dedicated function**.
- Software written for embedded systems is often called **firmware** and is stored in flash memory chips.
- Examples of embedded systems: MP3 players, traffic lights, digital watches
Outline

1.1 Computer hardware organization

1.2 The processor

1.3 Memory system operation

1.4 Program Execution

1.5 HCS12 Microcontroller
Semiconductor Memory

- Widely used in embedded systems.
- Non-volatile memory does not lose the information when there is no power, but volatile memory does.
- Read only memory: the MP can read but cannot write.
- Reading memory is nondestructive – memory does not lose data when CPU reads.
- Writing memory is destructive – when the CPU writes data to memory, the old data is written over and destroyed.

1- Random Access Memory (RAM)

- Read/write by MP.
- Same amount of time is required to access any location on the same chip.
- Dynamic random access memory (DRAM), and Static random access memory (SRAM) are volatile.
Magneto RAM (MRAM) and Ferroelectric RAM (FRAM) are non-volatile.

2- Read-only memory (ROM):

- Nonvolatile.

- Mask-Programmed ROM (MROM) is programmed once during manufacture.

- Programmable read-only memory (PROM): programmed once by the end user by using a special device.

- Erasable programmable ROM (EPROM): Electrically programmable many times by user by using a special device. The whole chip can be erased by ultraviolet light.

- Electrically erasable programmable ROM (EEPROM): Electrically Programmable many times by user by using a special device. Electrically erasable by using a special device. can be erased one location, one row, or whole chip in one operation.
Flash memory:
- Electrically erasable and programmable many times by the MP.
- Can erase a block or the whole chip.
- The most widely used nonvolatile memory technology.
- Most microcontrollers use on-chip flash memory as their program memory.
Memory System Operation

- Each memory location has two components: address and contents.

- The organization of a memory chip is indicated by $m \times n$, where $m$ is the number of locations in the chip and $n$ indicates the number of bits in one location.

- The number of address bits needed for selecting a memory location is $\log_2 m$

- For example, a memory chip has 16 bits addressing bits, how many locations are in this memory? $2^{16} = 64$ Kbytes

- How many addressing bits are needed to point to 64 Kbytes locations? $\log_2 64K = 16$ bits

- The memory data bits are connected to the data bus.

- The memory address bits are connected to the address bus.
- Registers are referred to by their names whereas memory locations are referred to by their addresses

Byte = 8 bits, Word = 2 bytes = 16 bits, Nibble = 4 bits

Kilobyte (KB) = $2^{10}$ bytes = 1026 bytes

Megabyte (MB) = $K^2$ Bytes = $2^{20}$ bytes = 1,048,576 bytes

Gigabyte (GB) = $K^3$ Bytes = $2^{30}$ bytes = 1,073,741,824 bytes
Read Operation

- RD (Read) in the MP is connected to OE (output enable) in the memory and WR (write) is connected to WE (write enable)
- The processor places the address of the memory location that it intends to read on the address bus and asserts the RD signal. (RD = 1 and WR = 0)
- After some time, the data stored in the location referred by the address will be out of the memory to the data bus to be read by the MP.
- RD and WR are two control signals to inform the memory to read or write.

![Diagram of a simplified memory system](image)

Figure 1.2 Block diagram of a simplified memory system
Write Operation

- The processor places the address of the location that it intends to write on the address bus.

- The processor also places the data that it intends to write on the data bus and asserts the WE signal (RD = 0 and WR = 1).

- The memory chip stores the data on the data bus in the selected memory location after some time.

- If a memory location is one byte but we want to store larger data, e.g., 16-bit number what should we do?

Store the data in two locations. To read the data, we read two locations and to write, we write two times.

- RD and WR can not be 1 at the same time but can be zeroes. Only one operation can be done at a time.
Outline

1.1 Computer hardware organization

1.2 The processor

1.3 Memory system operation

1.4 Program Execution

1.5 HCS12 Microcontroller
- In embedded systems, the program is stored in a nonvolatile memory.

- Where does the processor start to execute program after power on?

  (1) From a fixed location such as address 0 (Atmel microcontroller), or (2) Fetch the starting address from a fixed location in the memory ($FFFE and $FFFF in HCS12)

- How does the processor update the program counter (PC)? Increment PC by the length of the instruction just executed.

- How flow control instructions (such as Jump) are executed? Set the PC with the address of the target instruction.

- Flip-flop can store one bit, e.g., D flip-flop

  When set = 0, Q = 1 at the first rising edge clock

  When reset = 0, Q = 0 at the first rising edge clock

  Otherwise: Q = D at the first rising edge clock (CLK)
- The PC is 16-bit register having 16 flip-flops, each flip-flop stores one bit

- A common building block for the PC circuit is shown below.

1- Whenever the power is turned on, the flip-flops are reset, $PC = 0000$ and the instruction at address $0000$ is the first to be executed.

![Figure 1.4 A simplified block diagram of the program counter (PC) of an 8-bit microcontroller](image)
2- If the instruction being executed is a conditional branch and the branch condition is true, the branch signal will be 1, the MUX1 output is the branch offset.

The new PC value = the current PC value + the branch offset.

Figure 1.4 A simplified block diagram of the program counter (PC) of an 8-bit microcontroller.
3- If the instruction being executed is a jump instruction, then PC = Jump target (or the target address)

- In Jump instruction, there is no condition and the jump is always taken.

Figure 1.4 A simplified block diagram of the program counter (PC) of an 8-bit microcontroller

To jump to the instruction at $3000, PC = $3000
3- If the instruction being executed is not a program flow control instruction, then

The new PC value = the old PC value + the instruction’s number of bytes

PC is incremented by 1 after each instruction byte is fetched.

Figure 1.4 A simplified block diagram of the program counter (PC) of an 8-bit microcontroller
1- The assembler converts the assembly program to machine code and stores it in the memory starting from $0000.

```
ld 0x20, #0
ld 0x21, #20
ld ptr, #0x2000
loop: ld A,@ptr
and A, #0x03
bnz next
inc 0x20
next: dbnz 0x21,loop
```
2- Each time the microcontroller is reset, it starts executing from location $0000 \rightarrow$ that means PC should be $0000$

3- Two phases to execute an instruction:-

3.1 Instruction fetch:

- Read the instruction from the memory and store it in the instruction register (IR)

- May need more than one reading operation if the instruction is more than one byte.

- Update PC to point at the next instruction. How? Increment PC each time a byte of the instruction is read

3.2 Instruction Execution

- The control unit decodes the instruction and issues appropriate control signals to execute it.

- It may need to read/write the memory.
Instruction  \texttt{ld }$20,\#0 \texttt{ (machine code 75 20 00)}

Place 0 in data memory at 0x20

1- **Fetch instruction:**
- The Processor places contents of PC (0) on the address bus to read the instruction to be executed.
- It reads the first byte (opcode) = $75$. PC is incremented to $0001$.
- From the opcode, the control unit recognizes that it needs to read the following two bytes after the opcode. It reads $20$ and $00$ and holds them in IR. PC is incremented after each read operation to be 3 to point at the next executed instruction.

2- **Execute the instruction:**
- Address bus ← $20$ and Data bus ← $00$
- WR =1 to write 0 into the memory location $20$. 

Initially PC = 0

PC = PC + 1 after reading the byte pointed by 0 (or 75)

Executing this instruction does not change PC, but jump instruction does.
Instruction  \texttt{ld $21,\#$14} (machine code \texttt{75 21 14})

- Similar to previous instruction.
- PC = 0x0006 and write $14 to memory location $20.

-------------------------------------------------------

Instruction  \texttt{ld ptr,\#$2000} (machine code \texttt{90 20 00})

- Load $2000 into the ptr register

1- Fetch instruction:
- The address bus $\leftarrow$ PC value (0x06), request to read.
- Opcode = $90$ and PC = $07$.
- From the opcode, we need to read the following two locations. Read $20$ and $00$ and PC = 0x09.

2- Execute instruction:

\texttt{ptr \leftarrow 2000}
The machine code of the program

Figure 1.7 Instruction 3—opcode read cycle
**Instruction  ld A,@ptr (machine code E0)**

- A = the memory contents pointed to by register ptr
- Why this machine instruction needs only one byte?? Because the operands are registers. The instruction does not have immediate values or memory locations.

1- Fetch instruction
   - Address bus ← PC value (0x09), request to read, increment PC by 1 to be 0x0A.

2- Execute instruction
   - Read the memory location pointed by ptr. Address bus ← ptr and request a read operation.
   - Store the returned value by the memory in register A.
**Instruction**  \texttt{and A,#$03} (machine code 54 03)

- And the value 0x03 with accumulator A

1- Fetch instruction
   - Reading two bytes and update PC to $0C.

2- Execute instruction
   - Perform an AND operation on the contents of A and the value 0x03 and store the result in A
**Instruction**  `bnz next` (machine code is 70 02)

- Branch if the result of the last operation is not 0

1- Fetch instruction

- Read two bytes where 02 is the branch offset. PC is updated to 0x0E.

2- Execute instruction

- If the result of the last instruction is zero, \(\text{PC} = \text{PC} + 02\), (the jump is taken)

- Else does nothing because PC points at the following instruction – the branch should not be take.

**Notice:** Some instructions takes more execution time because they do more operations. Usually the instructions that need memory read/write takes more time.
Instruction **inc $20** (machine code 05 20)

- Increment the memory location at $20 by 1

1- Fetch instruction

   - Read two bytes PC is updated to $10F.

2- Execute instruction

   - Read the content of the memory location $20 and store in a MDR register.

   - Add one to the MDR register.

   - **Store MDR in memory location $20**: address bus ← $20, data bus ← content of MDR, and requests write operation.
Outline

1.1 Computer hardware organization

1.2 The processor

1.3 Memory system operation

1.4 Program Execution

1.5 HCS12 Microcontroller
The HCS12 Features

- 16-bit CPU (the operations can be done on 16 bit data)
- Maximum operating clock frequency 25 MHz
- 0 KB to 4KB of on-chip EEPROM
- 2KB to 14KB of on-chip SRAM
- 32 KB to 512KB on-chip flash memory
- Timer
- Serial communication interfaces
- 10-bit A/D converter
- I/O pins to interface to I/O devices
Two types of registers:-

1- **CPU registers**: used to perform general purpose operations such as arithmetic, logic, and program flow control.

2- **I/O registers**: used to configure the operations of the I/O devices and to hold data transferred in and out of the devices.

- CPU registers do not occupy memory space.
- I/O registers have addresses and are treated as memory locations when they are accessed.

**CPU registers**:

- Some registers are 8 bits others are 16 bits.

1- **General-purpose accumulators A and B**
   - Both A and B are 8 bits.
   - Most arithmetic functions are done on them.
   - A and B can be concatenated to form a single 16-bit register
     accumulator referred as the D accumulator.
The HCS12 CPU Registers

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit accumulator A and B or 16-bit double accumulator D</td>
<td></td>
</tr>
<tr>
<td>Index register X</td>
<td></td>
</tr>
<tr>
<td>Index register Y</td>
<td></td>
</tr>
<tr>
<td>Stack pointer</td>
<td></td>
</tr>
<tr>
<td>Program counter</td>
<td></td>
</tr>
</tbody>
</table>

Condition code register:
- Carry
- Overflow
- Zero
- Negative
- I Interrupt mask
- X Interrupt Mask
- Stop Disable

Figure 1.10 HCS12 CPU registers.
When A changes B is unaffected and vice versa, but when D changes both A and B change.

2- **Index registers X and Y**: 16 bits. Used to hold addresses. They are also used in several arithmetic instructions.

3- **Stack pointer (SP)**: 16 bits. A stack is a first-in-first-out data structure. SP points to the top byte of the stack as shown in figure.

4- **Program counter (PC)**: 16 bits holds the address of the next instruction to be executed.

5- **Condition code register (CCR)**: 8-bit register used to

   1- Flags are automatically updated after executing each instruction to reflect the status of the result.

   2- Enable and disable interrupts
Thank You!

Questions

Mohamed Mahmoud