Lecture 3:
Introduction to HCS12/9S12

Today's Topics

• Major pieces of information about the processor
• Specific information on MC9212DG256 microcontroller
Register Set

Programming Model

<table>
<thead>
<tr>
<th>Register Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit accumulators A &amp; B</td>
<td>A one-byte (8-bit) general purpose register. Since many mathematical operations can be performed using A, it is also referred to as the A accumulator.</td>
</tr>
<tr>
<td>16-bit accumulator D</td>
<td>A two-byte (16-bit) general purpose register. The D register is actually the concatenation of the A and B registers. A is used as the more significant byte with B as the less significant byte. Note, the two bytes worth of registers may be used as either A and B or as D, but not both at the same time.</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>
<tr>
<td>Condition Code Register</td>
<td>S X H I N Z V C</td>
</tr>
</tbody>
</table>

- The register set is also called the **programming model** of the computer.
- Programming Model
  - An abstract model of the microprocessor registers
  - This provides enough detail to understand the fundamentals of programming.
- In many processors, data may only be operated on if it is in a register.
Registers

Index Registers and Others

- X
  - A two-byte (16-bit) register primarily used to hold addresses. Very few mathematical operations can.

- Y
  - A two-byte (16-bit) register primarily used to hold addresses. Very few mathematical operations can.

- SP
  - A two-byte (16-bit) register used to manipulate the stack data structure.

- PC
  - Called the program counter, this is a two-byte (16-bit) register that holds the address of the next instruction to be executed.

- CCR
  - The condition code register maintains general operating status of the processor and some information used for branching. This one-byte register is the concatenation of eight 1-bit signals.

Memory Model

The way in which the microcomputer stores data

- Programmers usually visualize memory as a bunch of sequential spaces.

- Each space has a unique address that is used to refer the location.

- Number of memory units
  - Remember the two different architectures: Princeton* and Harvard

- Bit size of each location:
  - The number of bits stored in each location

- Bit size of the address
  - The number of bits used for the address limits the number of memory location
Endianness

Gulliver’s Travels

Big-endians crack soft-boiled eggs at the big end, and little-endians crack them at the other end in the story.

A microprocessor may need to store a number that is larger than a single memory location (in the HCS12, the size of memory location is 1 byte).

How to store 16-, 32- or 64-bit word to 8-bit address space.

Endianness means which byte is put first into the memory!
- Big-endian: put the big number portion of the large number first into the memory.
- Little-endian: put the little number portion of it first into the memory.

Big and Little-endian

Big-endians crack soft-boiled eggs at the big end, and little-endians crack them at the other end in the story.

A microprocessor may need to store a number that is larger than a single memory location (in the HCS12, the size of memory location is 1 byte).

How to store 16-, 32- or 64-bit word to 8-bit address space.

Endianness means which byte is put first into the memory!
- Big-endian: put the big number portion of the large number first into the memory.
- Little-endian: put the little number portion of it first into the memory.

Big and Little-endian
Endianness

Example

<table>
<thead>
<tr>
<th>Big-Endian</th>
<th>Little-Endian</th>
</tr>
</thead>
<tbody>
<tr>
<td>1FFF</td>
<td>1FFF</td>
</tr>
<tr>
<td>2000</td>
<td>2000</td>
</tr>
<tr>
<td>2001</td>
<td>2001</td>
</tr>
<tr>
<td>2002</td>
<td>2002</td>
</tr>
<tr>
<td>12</td>
<td>34</td>
</tr>
<tr>
<td>34</td>
<td>12</td>
</tr>
</tbody>
</table>

- The number 1234h stored at address 2000h

Type of Memory

- The memory map for the S12 which has **16-bit addresses** and **8-bit locations**.
- Different ranges of addresses are mapped to different types of storage.

| 0000h – 03FFh  | 0400h – 0FFFh  |
| 1000h – 3FFFh  | 3C00h – EF8Bh  |
| EF8Ch – EFFFFh | F000h – FFFFFh |
Instruction Set

- A list of all the operations that a processor can perform.
- A small section of the HCS12 instruction set.

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Operation Addr.</th>
<th>Mode</th>
<th>Machine Coding</th>
<th>Access Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Source Form:
  - Assembly code for the instruction

- Operation
  - A brief description that explains what the instruction does.

- Addressing mode
  - It tells how the instruction uses the operand(s), if any.

• Machine Coding
  - The hexadecimal value that represents the instruction in memory.
  - Also called the instruction format since it shows how to convey the operation and its operands to the processor.

• Access Detail
  - Each letter stands for the internal operation performed during each clock cycle required by the operation.
  - The number of letters = the number of clock cycles taken.

• SXHINZVC: Condition Code Register
  - Δ: affected by operation, 1: set 1, and 0: set 0 after the instruction.
Programming Flow

High Level Code

- C, C++, Basic, Pascal, Fortran, and others
- Usually exist as a text file.
- A portion of high level may be written without regard to the specific processor that will eventually run the program
- A **compiler** converts high level code to assembly code that runs on the same processor as the compiler runs
- A **cross-compiler** runs on one type of processor and converts high level code to assembly for a different type of processor.
- High level languages do not have instructions that can access all of a microcomputer’s instructions. Many programs written mainly in a high level language have sections of assembly code.
- One line in a high level language may compile into several, possibly hundreds, of lines of assembly.

Programming Flow

Assembly Code

- A somewhat “human readable” form of the exact code that will be executed on the processor
- Usually exists as a text file
- An assembler converts assembly code to machine code that runs on the same processor as the assembler runs
- A **cross-assembler** runs on one type of processor and converts assembly code to machine code for a different type of processor.
- Assembly code itself is not executed
- Assembly code is specific to a given type, or family, of processors.
- Each line of assembly code uniquely corresponds to one instruction in machine code.
Programming Flow

Machine Code

- The string of 1’s and 0’s representing the operations.
- The exact values that are loaded by the microprocessor from memory to execute the program.
- On PCs, these are executable (often .EXE) files.
- May not be executed on other types of microprocessors

Questions?
Wrap-up
What we’ve learned

- Registers – Programming Model
- Memory Model – Endianness
- Programming flow

What to Come

- Addressing Mode!