

# Introduction to HC11

N. Narasimhamurthi

## 1 Objective

To become familiar with HC11 and using BUFFALO utilities, interacting using the terminal program, transferring files, editing-assembling-loading programs and executing them.

### 1.1 Tasks

#### 1.1.1 Getting started with HC11

1. Disconnect power from the 68HC11
2. Connect 68HC11 to your computer using the serial cable that came with the machine.
3. Start the terminal program Hyperterm. The program is already configured to communicate with the 68HC11.
4. Power up 68HC11

You should see the following prompt

```
BUFFALO 2.5 (ext) - Bit User Fast Friendly Aid to Logical Operation
>
```

Type h (for help) and press the enter key. You should see the following help screen

```
ASM [<addr>] Line assembler/disassembler.
/          Do same address.          ...
CTRL-J    Do next address.          ...
CTRL-A    Quit.
BF <addr1> <addr2> [<data>] Block fill.
BR [-]<addr> Set up breakpoint table.
...
...
```

### 1.1.2 Looking at memory

To see what is stored in the HC11 memory, we use the memory dump command. The instruction for dumping memory is MD. Let us look at what is present in locations E000 to E3FF. At the prompt, enter the command MD E000 E3FF. You should see:

```
>MD E000 E03F
```

```
E000 CE 10 0A 1F 00 01 03 7E B6 00 86 93 B7 10 39 86          9
E010 00 B7 10 24 8E 00 68 BD E3 40 CE 00 4A DF A7 86      $ h @ J
E020 D0 97 A6 CC 3F 0D DD 69 BD E1 9A 7F 00 A9 7C 00      ? i
E030 A9 7F 00 AB B6 10 3C 84 20 27 35 86 03 B7 98 00      < '5
>
```

Each line of output consists of three parts. First there is the memory address, for example E020. This is followed by 16 bytes of data. These are contents of 16 locations starting from the address. For example, in the above example, memory location E020 contains D0, location E021 contains 97 etc. Recall that all numbers are in hex. After the 16 bytes come 16 characters. Each of the bytes is interpreted as an ASCII code and the character that the code represents is shown. Non printable and non-ascii characters are shown as a space.

#### Exercise

1. Look up the ascii code for the letter J. Can you locate it in the memory dump shown above?
2. Determine the contents of the locations FFD0 to FFFF. Write down the contents in your lab notebook. You will need these values later in the course.

### 1.1.3 Modifying memory

Now that you know what is in the memory, let us try to modify the contents of the memory. Let us store the following values in locations starting from 2100: B6, 30, 00, BB. To modify memory, we use *Memory modify* command, MM. When you modify the memory, the command will first echo the value already in memory. If you don't want to change it, press the space bar, command will move on to the next memory location. If you want to change the value, just type the value (in HEX). If you make a mistake, **do not press the backspace key**. Continue typing. MM will only look at the last two characters you typed. So if you type AB872B, you have in effect typed 2B. Once you have entered the correct value, press the space bar. When you are all done, press the enter key. Here is a quick list to remind you:

1. To leave memory unaltered but to move to next location, press SPACE bar
2. To modify memory and then move to next location, enter the data and then press the SPACE bar
3. When entering the value, only the last two characters are used. So if you make a mistake continue typing
4. When done, press the enter key

The following shows the interaction for entering the four values. Since **MM** echoes the previous values stored in memory, you may see different numbers:

```
>MM 2100
2100 A7 B6 BB 30 C2 00 32 BB
>
```

After every memory modify command get into the habit of running the memory dump command to make sure that the memory was modified the way you wanted.

### Exercise

1. Try modifying memory at location **E000**. What happens when you do it?
2. Modify the memory at three locations starting from **3000** to the following values **10 32 A8** and make sure that the changed did take place.
3. Turn the power to HC11 (not the PC!) off and then on again. What happened to the changes you made to locations **3000** to **3002**? Why?
4. Look at memory locations **E000** to **E00F**. Did any of them change when you cycled the power?

#### 1.1.4 Writing and entering your first program: Using MM

Let us write a short program that will add three numbers stored in locations **3000**, **3001** and **3002**. This will be done in five steps

1. First we will load the register **A** with the value stored in location **3000**
2. Next we will **add to** the register **A** the value stored in location **3001**
3. Next we will **add to** the register **A** the value stored in location **3002**

4. Next we will store the value in the register **A** in location 3003
5. We will return control back to BUFFALO.

We consult the little pink book and find that the command to load a value into register **A** is called **LDAA**. Since we want to load from memory, we need the **EXT**<sup>1</sup> mode of the command. The code for the command, commonly known as the operation code, or more simply **OPCODE**, is **B6**. So our first instruction is **B6 30 00**. We say that the instruction is **LDAA 3000**. Similarly, we find that the second instruction is **BB 30 01** which corresponds to **ADDA 3001**. The command for storing data is called **STAA** and the **OPCODE** is **B7**. The instruction to return control back to BUFFALO is **3F** and is called **SWI**. Thus the entire program is:

**B6 30 00 BB 30 01 BB 30 02 B7 30 03 3F**

Now that you have written the program, you have to load it into HC11 memory. You do that using the memory modify command. Before you can do that, you have to decide *where* you want to store the program. Check with your TA to see if he has a preferred location<sup>2</sup>. For now, we will use the locations starting at **2100**. So type the command **MM 2100** and enter your program, one byte at a time.

### 1.1.5 Running your first command

First modify the locations 3000-300f to values you can easily recognize (but not all zeros). Assuming that you stored the program starting at location **2100**, you run the program by using the call command

```
call 2100
```

Note you can abbreviate any command by entering enough characters to identify it. Since no other command starts with the letter **c**, you could also have typed **c 2100**<sup>3</sup>. When the control is returned to BUFFALO, it prints the contents of all the registers and you should see something like:

```
>CALL 2100
P-210C Y-AAAA X-AAAA A-EA B-AA C-D8 S-004A
>
```

---

<sup>1</sup>At this stage other modes will not make sense. You will soon learn about the other modes

<sup>2</sup>You can pretty much enter your program anywhere you have RAM. However, to make it easy for the TA when he goes from one student to the next, each TA may state certain standard locations

<sup>3</sup>An alternative to call is the go command, **G**. There is no difference between them if your program ends with **SWI**. However, if your code ends with an **RTS**, you have to use the call command.

Do a memory dump of locations 3000–300F. You should see something like:

```
>md 3000 300f
3000 10 32 A8 EA FF FF FF FF FF FF FF FF FF FF FF  2
>
```

## Exercises

1. Modify the values in locations 3000–300f and run your program. Write down the values in the locations 3000–3003 and the value in register **A**, after you run the program.
2. Repeat the above 5 times and explain your results.

### 1.1.6 Entering your program: Using ASM

We will reenter your program, except we will store it at a different location. Check with your TA for the desired location. If he does not, enter the program starting at address **2200**, using the asm command. When you enter the ASM command, BUFFALO will display instruction currently stored in memory (or a question mark if it is not a valid instruction). You can press enter if you want to leave memory unchanged or type the command and then press the enter key. On power-up, the BUFFALO performs a memory check and fills all of RAM with **FF**. **FF** happens to be the code for **STX**. So ASM command will often display **STX \$FFFF**<sup>4</sup>. Here is how I entered the program at location 2200. What I typed is shown in **boldface**:

---

<sup>4</sup>BUFFALO expects all its data to be entered in HEX. So you do **NOT** type the \$ in front of numbers to indicate that the number is entered using HEX representation. However, BUFFALO adds the \$ in front of the numbers written using HEX representation

---

```

>ASM 2200
2200      STX $FFFF
          >LDAA 3000
          B6 30 00
2203      STX $FFFF
          >ADDA 3001
          BB 30 01
2206      STX $FFFF
          >ADDA 3002
          BB 30 02
2209      STX $FFFF
          >STAA 3003
          B7 30 03
220C      STX $FFFF
          >SWI
          3F
220D      STX $FFFF
          > CTRL-A

```

---

Note that to get out of the ASM command you need to type the control-A character.

### Exercises

1. Enter the program shown above and verify that the program is entered correctly.
2. Enter the following program starting at location 2300 and run it. What is the result of running your code?

```

LDX #E600
JSR FFC7
SWI

```

3. (a) Modify the above program by changing E600 to 3200.
  - (b) Find out the ascii codes for the letters in the phrase **THIS CLASS IS FUN**. To get you started, here are some codes: code for 'T' is 54, code for 'H' is 48 and code for I is 49.
  - (c) Enter the ASCII codes starting from location 3200. After you enter the last ASCII code, enter the special code **04**.

- (d) Run the program and write down what output you get.
- (e) To see why you need the special code, do memory modify and change it to 0A. Rerun the program and write down what you observe.

### 1.1.7 Entering your program: Using assembler

In this method, we will do all the work on the PC and eventually transfer the program to HC11. Start any text editor. **DO NOT USE A WORDPROCESSOR** such as Word, Wordpad etc. The best editor I am aware of the Programmer's file editor (PFE). This editor is part of the lab package that you can download from the web. Create a directory/folder where you will do all your work. In that directory, copy all the files from the lab pack.

Using the editor, create a file called `PROG1.ASM` and enter the following<sup>5</sup>:

```
*Name: your name goes here
*Uniqname: Your unique name goes here
*Class: ECE 373 (or your course number)
*Term: The term
*Date: Date you started the code
  ORG $2500 *this determines where the code will be stored
  LDAA $3000
  ADDA $3001
  ADDA $3002
  STAA $3003
  SWI
```

Save your file. Start a MSDOS. Change directory to where your files are stored. Execute the command

```
ASM PROG1
```

If there are no errors, you should see two new files, `PROG1.LST` and `PROG1.S19`. The first file for humans to read and is often called the LST file and the second is for the HC11 and is called the S19 file.

Go back to the BUFFALO prompt and type the command

```
LOAD T
```

In Hyperterm, use the ASCII file transfer command (use the Transfer menu item to get to it) to transfer the `PROG1.S19` file. When the transfer is complete, you should see the program in locations starting from 2500.

---

<sup>5</sup>By tradition, assembly language programs are written in UPPERCASE letters, except if you are writing code for Unix and its derivatives.

**Exercises:**

1. Write the above code, assemble it (i.e. create the S19 file), transfer the S19 file to HC11 and run the program. Verify that the program works correctly.
2. Copy the file LAB1.ASM. The file has three errors in it. Assemble it and look at the error messages. Fix the errors and submit a corrected version.