CS 255 **Homework 4 – Addressing Modes**

Due: Th. Feb. 19th by the start of class

This is an individual assignment. See the class webpage for appropriate collaboration policies and the statement you need to include with each assignment (as comments at the top of your file). Failure to include your collaboration statement will result in a -10 deduction.

Background and Setup:
The purpose of this homework is to reinforce the 5 different addressing modes we discussed in class and operand sizes for instructions.

I have provided a skeleton program and debug file to help you get started. Make a `hw4` subdirectory in your `cs255` directory. Copy the files from the `~cs255000/share/hw4` directory to your `~/cs255/hw4` directory. You should see 3 files: `hw4.s, egt.hw4.pc` and `egt.hw4.mac`.

- If you are working on a physical lab machine, or via an `ssh -X` command, execute the following at the command line in your hw4 directory on the Math/CS system:
  `rm egt.hw4.pc; mv egt.hw4.mac .egt.hw4`
- If you are working via the m68000-server setup, you will need to put either the `egt.hw4.pc` or the `egt.hw4.mac` file on YOUR machine in the same directory as you have placed the `tunnel`, `m68000` and `Egtapi.jar` files. Download the appropriate file from the class website homework page for your operating system (either PC or Mac/Linux) to YOUR computer and rename the file to `.egt.hw4`. You MUST use the appropriate file for your operating system!

Assignment:
Similar to HW3, the skeletal program has 10 sections labeled `Q1, Q2, … Q10` corresponding to the answer to each question. You should write the assembly language instructions to answer each question in the appropriate, corresponding section. DO NOT remove or change the `Q1...Q10` labels. Each question will ask you to get values from arrays (see #3 and #4 below) and move/copy the values into an answer variable, `ans_b, ans_w`, or `ans_l`

The variable section of the assembly program contains:

1. variables used to display output: `ans_b` (one byte), `ans_w` (one word) and `ans_l` (one longword)
2. indexes used to access array elements: `i` (byte), `j` (word), and `k` (longword). These are currently initialized to 3, 4, and 5 respectively.
3. arrays of 10 elements each: `A` (byte array), `B` (word array), and `C` (longword array)
4. a linked list starting at the variable `head`. This linked list contains elements which contain TWO pieces of data and a pointer to the next item, as defined in this class:

```
class list {
   int value1;
   short value2;
   list next;
}
```

Running and Testing Your Program:
The `.egt.hw4` file is read in by the `m68000` program (EGTAPI) and instructs `m68000` to stop at the listed "breakpoints" (`Start, Stop, End, Q2, Q3, ..., Q9`), and to display various variables in the program (`i, j, ans_l`, etc). You can take a look at the `.egt.hw4` file, but you do not need to change it. It will be loaded automatically for you when you load the hw4 program (below).

Questions:
Translate each of the following Java statements into assembly language at the correct label in the hw4.s file. Be careful to always use the correct operand size for every instruction.

Remember the following equivalences:

Java Datatype	Size
`byte`	1 byte / 8 bits
`short`	2 bytes / 16 bits
`int`	4 bytes / 32 bits

1. `ans_b = A[5];` (ans: -66)
2. `ans_l = B[4];` (ans: 555)
3. `ans_l = C[k];` (ans: -6666)
4. `ans_w = A[j + k];` (ans: -123)
5. `ans_w = C[i + k];` (ans: 9999)
6. `ans_l = A[j] + B[i];` (ans: 55 + -444 = -389)
7. `ans_l = A[A[j] – 50];` (ans: A[55 - 50] = A[5] = -66)
8. `ans_w = B[13];` (Figure out for yourself what happens)
9. `ans_l = head.value1;` (ans: 1234)
10. `ans_w = head.next.next.value2;` (ans: 67)

Running and Testing Your Program:

1. Assemble your hw4.s program using the command

   `as255 hw4`

2. Start EGTAPI with the `m68000` command and load the hw4 program into EGTAPI.

3. In this homework, you will want to carefully watch the variables `ans_b, ans_w`, and `ans_l` in the "Variable Watch" window on the right hand side of EGTAPI. You will use this display to verify you have solved a question correctly.

4. The "Continue" button (2 right arrows, next to the "Single Step" button) will run your program until a breakpoint is encountered. For this homework, breakpoints have been added at every question label, `Q1 – Q10` via the `.egt.hw4` file. You can add additional breakpoints by adding labels to your `hw4.s` program and then adding that breakpoint and clicking the "Add" button in the Breakpoint window in EGTAPI (lower right hand corner).

5. Program the answer to Question 1. Assemble and re-load the file into EGTAPI. When you click the "Continue" button, the program will run until the first breakpoint, which is set at the `Q2` label. In other words, the code answering Q1 will have executed when the program pauses/stops at label Q2. You should see the variable `ans_b` with the value -66 in the variable watch window. If you do not, Q1 is not correct and you should reload and single step through your instructions to find your error.

6. Note that the correct answers you should see are given for all questions. Use this as a learning opportunity to carefully check your understanding of addressing modes and operand sizes. Don't just blindly try things until you arrive at the correct answer via sheer luck. I recommend working on each question, one at a time rather then trying to write code for all of them and test them all at once.

7. When you have completely run the program and see that the next instruction is a `nop` instruction, **do NOT** attempt to run further (this will cause unpredictable errors).

Debugging Help:

Let m68000 help you debug the program as much as possible. The following are general guidelines for debugging assembly programs:

1. Run m68000 and get to the part of the program containing a bug. You can set a breakpoint and use the "Continue" button to get you to the trouble spot quickly.

2. When you have reached the place with a bug, use the "Single Step" button to execute each instruction and evaluate if the result is indeed what you have intended.

Example: Your answer for question Q6 is wrong.

- Start m68000 and load your program if you have not already done so.
- Press "Continue" until you have reached the breakpoint labeled Q6 which is the address of the first instruction involved in answering question #6.
- Now use the "Single Step" button to execute one instruction.
- m68000 will display the outcome of the execution of each instruction and you can find the bug by verifying your understanding of the instruction (ie, what you *think* it will do) against the state of the machine displayed in EGTAPI (ie, what it *actually* does). (Technically, you need to

know what you want the instruction to achieve so you know what to look for. Thus, an essential part of "debugging" requires that you know what you want to achieve.)

- Note that you can always remove breakpoints if you want to. Simply type in the breakpoint name (ex: Q4) and then click remove in the Breakpoints window.

Submission:

Issue the submit command from any of the computer lab machines while you are in your `~/cs255/hw4` directory:

```
/home/cs255000/turnin hw4.s hw4
```

You will see an "ALLOWED" message if your submission was submitted correctly. If you do not see this message, make sure you are issuing the command from your `/home/yourID/cs255/hw4` directory and that you do not have any typos in your command. If you are still unsuccessful, contact Dr. Summet for assistance.