CS 218 – Assignment #2

Purpose: Become familiar with the assembler, linker, and debugger. Learn how to display values in memory for integers, reals, and characters.
Due: Thursday (6/09)
Points: 35 (Part A - 20 pts, Part B - 15 pts)

Assignment:

Part A:
Write a simple assembly language program to compute the following formulas:

```
bAns1  =  bVar1 + bVar2
bAns2  =  bVar1 - bVar2
wAns1  =  wVar1 + wVar2
wAns2  =  wVar1 - wVar2
dAns1  =  dVar1 + dVar2
dAns2  =  dVar1 - dVar2
```

Declare the following variables in the data segment (after the ".data").

```
NULL            equ    0

bVar1           db     38
bVar2           db     16
bAns1           db     0
bAns2           db     0
wVar1           dw     3716
wVar2           dw     1867
wAns1           dw     0
wAns2           dw     0
dVar1           dd     168318283
dVar2           dd     135678291
dVar3           dd     -47156
dAns1           dd     0
dAns2           dd     0
flt1            dd     -13.25
flt2            dd     -14.125
fourPi          dd     12.56636
qVar1           dq     134278427262
myClass         db     "CS 218", NULL
edName          db     "Ed Jorgensen", NULL
myName          db     "your name goes here", NULL
```

Edit the provided main, including you name/section in the comments, add the data (from above) updating you name for the *myName* variable, and the code from class.

Part B:
Complete the Assignment #2 Data Representation Worksheet. The answers for the worksheet can be obtained using the debugger once the program is completed.

Submission:

- All source files must assemble and execute on Ubuntu with **`yasm`**.
- Submit source files
  - Submit a copy of the program source file via the on-line submission
- Once you submit, the system will score the project and provide feedback.
  - If you do not get full score, you can (and should) correct and resubmit.
  - You can re-submit an unlimited number of times before the due date/time.
- Late submissions will be accepted for a period of 24 hours after the due date/time for any given lab. Late submissions will be subject to a ~2% reduction in points per an hour late. If you submit 1 minute - 1 hour late -2%, 1-2 hours late -4%, … , 23-24 hours late -50%. This means after 24 hours late submissions will receive an automatic 0.

Program Header Block

All source files must include your name, section number, assignment, NSHE number, and program description. The required format is as follows:

```
;  Name: <your name>
;  NSHE ID: <your id>
;  Section: <section>
;  Assignment: <assignment number>
;  Description: <short description of program goes here>
```

Failure to include your name in this format will result in a loss of up to 5%.

Scoring Rubric

Scoring will include functionality, code quality, and documentation. Below is a summary of the scoring rubric for this assignment.

Criteria	Weight	Summary
Assemble	-	Failure to assemble will result in a score of 0.
Program Header	5%	Must include header block in the required format (see above).
General Comments	10%	Must include an appropriate level of program documentation.
Program Functionality (and on-time)	85%	Program must meet the functional requirements as outlined in the assignment. Must be submitted on time for full score.

Debugger Commands:

Execute the program in the debugger (in the same manner as assignment #1). You should review the DDD/GDB debugger information handout to understand the debugger commands examine memory variables.

You should use the provided "**a2in.txt**" to display the variables with the debugger.

- Each byte, word, double-word sized, and quadword variable is displayed twice (once in decimal and again in hex).
- The floating point values are display twice (once as a real value and again in hex).
- The strings are displayed twice, once showing both the decimal and ASCII values and then just the hex values for the first six characters

A brief summary of the command to examine memory is as follows:

x/<n><f><u> &<variable>	Examine memory location <variable>		
	<n>	number of locations to display, 1 (number one) is the default.	
	<f>	format:	d – decimal
			x – hex
			u – unsigned
			c – character
			s – string
			f – floating point
	<u>	unit size:	b – byte (8-bits)
			h – halfword (16-bits)
			w – word (32-bits)
			g – giant (64-bits)

For example, to display the 16-bit variable **wVar2** and the 32-bit variable **dVar1**, and the 64-bit variable **qVar1**, the commands would be as follows:

```
x/dh &wVar2
x/dw &dVar1
x/dg &qVar1
```

For future assignments you will need to select the correct command to display the data based on the defined size and any guidance from the assignment.

More detailed information can be found in Chapter 6 of the class text.