CS 218 - Assignment #3

Purpose:	Become familiar with the assembler, linker, and debugger. Display values in memory and learn to use basic arithmetic instructions.
Due:	Tuesday (2/05)
Points:	40

Assignment:

Use the provided assembly language program template to compute the following calculations:

```
; ****************************************
; Byte Operations
; unsigned byte additions
;         bAns1  = bNum2 + bNum3
;         bAns2  = bNum1 + bNum3
;         bAns3  = bNum4 + bNum3
; -----
; signed byte additions
;         bAns4  = bNum5 + bNum7
;         bAns5  = bNum6 + bNum8
; -----
; unsigned byte subtractions
;         bAns6  = bNum1 - bNum3
;         bAns7  = bNum2 - bNum3
;         bAns8  = bNum3 - bNum4
; -----
; signed byte subtraction
;         bAns9  = bNum5 - bNum8
;         bAns10 = bNum7 - bNum6
; -----
; unsigned byte multiplication
;         wAns11  = bNum1 * bNum4
;         wAns12  = bNum2 * bNum2
;         wAns13  = bNum3 * bNum2
; -----
; signed byte multiplication
;         wAns14  = bNum5 * bNum6
;         wAns15  = bNum7 * bNum8
; -----
; unsigned byte division
;         bAns16 = bNum1 / bNum3
;         bAns17 = bNum4 / bNum2
;         bAns18 = wNum3 / bNum4
;         bRem18 = wNum3 % bNum4
; -----
; signed byte division
;         bAns19 = bNum5 / bNum8
;         bAns20 = bNum6 / bNum7
;         bAns21 = wNum5 / bNum8
;         bRem21 = wNum5 % bNum8

; ****************************************
; Word Operations
; -----
; unsigned word additions
;         wAns1  = wNum4 + wNum1
;         wAns2  = wNum3 + wNum2
;         wAns3  = wNum3 + wNum3
```

```
; -----
; signed word additions
;         wAns4   = wNum5 + wNum7
;         wAns5   = wNum6 + wNum8
; -----
; unsigned word subtractions
;         wAns6   = wNum1 - wNum3
;         wAns7   = wNum2 - wNum4
;         wAns8   = wNum4 - wNum2
; -----
; signed word subtraction
;         wAns9   = wNum5 - wNum8
;         wAns10   = wNum6 - wNum7
; -----
; unsigned word multiplication
;         dAns11 = wNum1 * wNum4
;         dAns12   = wNum4 * wNum4
;         dAns13   = wNum2 * wNum3
; -----
; signed word multiplication
;         dAns14   = wNum5 * wNum7
;         dAns15   = wNum6 * wNum8
; -----
; unsigned word division
;         wAns16 = wNum2 / wNum3
;         wAns17 = wNum1 / wNum4
;         wAns18 = dNum1 / wNum2
;         wRem18 = dNum1 % wNum2
; -----
; signed word division
;         wAns19 = wNum5 / wNum8
;         wAns20 = wNum7 / wNum6
;         wAns21 = dNum5 / wNum7
;         wRem21 = dNum5 % wNum7

; ***************************************
; Double-Word Operations
; -----
; unsigned double-word additions
;         dAns1   = dNum1 + dNum4
;         dAns2   = dNum2 + dNum3
;         dAns3   = dNum3 + dNum4
; -----
; signed double-word additions
;         dAns4   = dNum5 + dNum8
;         dAns5   = dNum6 + dNum7
; -----
; unsigned double-word subtractions
;         dAns6   = dNum1 - dNum4
;         dAns7   = dNum2 - dNum3
;         dAns8   = dNum3 - dNum4
; -----
; signed double-word subtraction
;         dAns9   = dNum5 - dNum7
;         dAns10 = dNum6 – dNum8
; -----
; unsigned double-word multiplication
;         qAns11   = dNum1 * dNum2
;         qAns12   = dNum3 * dNum4
;         qAns13   = dNum2 * dNum3
; -----
; signed double-word multiplication
```

```
;         qAns14  = dNum5 * dNum6
;         qAns15  = dNum7 * dNum8
; -----
; unsigned double-word division
;         dAns16 = dNum1 / dNum4
;         dAns17 = dNum2 / dNum3
;         dAns18 = qAns12 / dNum4
;         dRem18 = qAns12 % dNum4
; -----
; signed double-word division
;         dAns19 = dNum5 / dNum8
;         dAns20 = dNum6 / dNum7
;         dAns21 = qAns12 / dNum8
;         dRem21 = qAns12 % dNum8

; ***************************************
; QuadWord Operations
; -----
; unsigned quadword additions
;         qAns1  = qNum1 + qNum3
;         qAns2  = qNum2 + qNum4
;         qAns3  = qNum2 + qNum4
; -----
; signed quadword additions
;         qAns4  = qNum5 + qNum8
;         qAns5  = qNum6 + qNum7
; -----
; unsigned quadword subtractions
;         qAns6  = qNum1 - qNum2
;         qAns7  = qNum3 - qNum4
;         qAns8  = qNum2 - qNum4
; -----
; signed quadword subtraction
;         qAns9  = qNum6 - qNum7
;         qAns10 = qNum5 - qNum8
; -----
; unsigned quadword multiplication
;         dqAns11  = qNum1 * qNum2
;         dqAns12  = qNum3 * qNum4
;         dqAns13  = qNum2 * qNum3
; -----
; signed quadword multiplication
;         dqAns14  = qNum5 * qNum8
;         dqAns15  = qNum6 * qNum7
; -----
; unsigned quadword division
;         qAns16 = qNum1 / qNum4
;         qAns17 = qNum2 / qNum3
;         qAns18 = dqAns12 / qNum2
;         qRem18 = dqAns12 % qNum2
; -----
; signed quadword division
;         qAns19 = qNum5 / qNum6
;         qAns20 = qNum7 / qNum6
;         qAns21 = dqAns12 / qNum8
;         qRem21 = dqAns12 % qNum8
```

Refer to the on-line text for information and examples of the addition, subtraction, multiplication, and division instructions.

Data Declarations:
Use the data declarations in the provided main.

Submission:
When complete, submit:

- A copy of the ***source file*** via the class web page (assignment submission link) by class time (Section 001, 4:00 PM and Section 002, 5:30 PM).
- Assignments received after the due date/time will not be accepted.

Debugger Commands
You will need to execute the code and display the variables in the same manner as previous assignments. The command to examine memory is as follows:

x/<n><f><u> &<variable>	Examine memory location <variable>		
	<n>	number of locations to display, 1 is defualt.	
	<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, some of the applicable memory examine commands for various data types are as follows:

Operation	**Command**
Display signed decimal byte values.	**`x/db &bnum1`**
Display unsigned decimal byte values.	**`x/ub &bnum1`**
Display signed decimal word values.	**`x/dh &wnum1`**
Display unsigned decimal word values.	**`x/uh &wnum1`**
Display hex word values.	**`x/xh &wnum1`**
Display signed decimal double-word values.	**`x/dw &wnum1`**
Display unsigned decimal double-word values.	**`x/uw &wnum1`**
Display hex double-word values.	**`x/xw &wnum1`**
Display signed decimal double-word values.	**`x/dg &wnum1`**
Display unsigned decimal double-word values.	**`x/ug &wnum1`**
Display hex quadword values.	**`x/xg &wnum1`**

You may use the provided "**`a3in.txt`**" to display the variables within the debugger. However, 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. Refer to the on-line text for additional information.