CS 218 - Assignment #3

Purpose: Become familiar with the assembler, linker, and debugger. Display values in memory and learn to use basic arithmetic instructions.

Points: 60

Assignment:

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

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

; *******************************************
; Word Operations
; unsigned word additions
;        wAns1  = wNum1 + wNum2
;        wAns2  = wNum1 + wNum3
;        wAns3  = wNum3 + wNum4
; -----
; signed word additions
;        wAns4  = wNum5 + wNum6
;        wAns5  = wNum5 + wNum3
; -----
; unsigned word subtractions
;        wAns6  = wNum2 - wNum1
;        wAns7  = wNum4 - wNum2
;        wAns8  = wNum1 - wNum3
; -----
; signed word subtraction
;        wAns9  = wNum5 - wNum6
;        wAns10 = wNum5 - wNum3
; -----
; unsigned word multiplication
;        dAns11 = wNum1 * wNum3
;        dAns12 = wNum3 * wNum4
;        dAns13 = wNum2 * wNum3
; -----
; signed word multiplication
;        dAns14  = wNum5 * wNum6
;        dAns15  = wNum6 * wNum1
; -----
; unsigned word division
;        wAns16 = wNum2 / wNum3
;        wAns17 = wNum4 / wNum1
;        wAns18 = dAns13 / wNum1
;        wRem18 = dAns13 % wNum1
; -----
; signed word division
;        wAns19 = wNum5 / wNum1
;        wAns20 = wNum5 / wNum3
;        wAns21 = dAns14 / wNum1
;        wRem21 = dAns14 % wNum1

; ********************************************
; Double-Word Operations
; unsigned double word additions
;        dAns1  = dNum2 + dNum3
;        dAns2  = dNum3 + dNum4
;        dAns3  = dNum2 + dNum4
; -----
; signed double word additions
;        dAns4  = dNum5 + dNum6
;        dAns5  = dNum5 + dNum2
; -----
; unsigned double word subtractions
;        dAns6  = dNum1 - dNum4
;        dAns7  = dNum4 - dNum3
;        dAns8  = dNum3 - dNum2
; -----
; signed double word subtraction
;        dAns9  = dNum5 - dNum6
;        dAns10 = dNum5 - dNum2
; -----
; unsigned double word multiplication
;        qAns11  = dNum2 * dNum2
;        qAns12  = dNum1 * dNum2
;        qAns13  = dNum2 * dNum4
; -----
; signed double word multiplication
;        qAns14  = dNum5 * dNum6
;        qAns15  = dNum5 * dNum2
; -----
; unsigned double word division
;        dAns16 = dNum1 / dNum4
;        dAns17 = dNum3 / dNum2
;        dAns18 = qAns13 / dNum2
;        dRem18 = modulus (qAns13 / dNum2)
; -----
; signed double word division
;        dAns19 = dNum5 / dNum2
;        dAns20 = dNum5 / dNum6
;        dAns21 = qAns15 / dNum4
;        dRem21 = qAns15 % dNum4

; *****************************************
; QuadWord Operations
; unsigned quadword additions
;        qAns1  = qNum1 + qNum2
;        qAns2  = qNum3 + qNum2
;        qAns3  = qNum3 + qNum1
; -----
; signed quadword additions
;        qAns4  = qNum6 + qNum7
;        qAns5  = qNum5 + qNum8
; -----
; unsigned quadword subtractions
;        qAns6  = qNum3 - qNum1
;        qAns7  = qNum4 - qNum2
;        qAns8  = qNum4 - qNum1
; -----
; signed quadword subtraction
;        qAns9  = qNum7 - qNum6
;        qAns10 = qNum8 - qNum7
; -----
; unsigned quadword multiplication
;        dqAns11  = qNum4 * qNum3
;        dqAns12  = qNum3 * qNum4
;        dqAns13  = qNum2 * qNum2
; -----
; signed quadword multiplication
;        dqAns14  = qNum6 * qNum7
;        dqAns15  = qNum5 * qNum8
; -----
; unsigned quadword division
;        qAns16 = qNum3 / qNum1
;        qAns17 = qNum4 / qNum3
;        qAns18 = dqAns11 / qNum3
;        qRem18 = dqAns11 % qNum3
; -----
; signed quadword division
;        qAns19 = qNum5 / qNum6
;        qAns20 = qNum8 / qNum7
;        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. *Note*, the main includes some of the calculations already done as examples.

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 assignment. 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

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 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, 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 text for additional information.