Assignment 3 Cover Page

Structure and Application of Microcomputers

Student Information (Print)

Conor
First Name

Roberts
Last Name

1056167
Student ID

Name of Marker: _____ **Mark:** _____

Instructions: This assessment is designed to emphasize and reinforce some of the important concepts related to reading, lecture, and lab materials covered in week 3. To maximize your benefit from this exercise you are encouraged to do the following:

(1) Read through the assignment questions *before* listening to the lectures.
(2) Answer as many questions as possible in tandem with viewing and listening to the lectures. You can easily pause a lecture to answer a question, or re-listen to any material that you do not fully understand.
(3) Simply writing down a solution to a particular problem is not enough to obtain full points. You must show how you arrived at the solution by showing the step-by step procedure that you performed. This procedure can be hand-written or typed – the choice is yours. However, make sure to answer each question in the provided spaces.
(4) Review your answers a few days after completing the assignment. Taking a few minutes to do this can dramatically improve your understanding of key concepts and increase your retention.

Submission: Submission of this assessment is a two-step process that involves (1) uploading your written work as a single PDF to the Dropbox on CourseLink, and (2) entering only the final answers to the assignment questions under the Quiz Section on CourseLink. **Only assignments that complete both steps will be marked. The overall submission procedure is outline below.**

(1) Once you have filled in the cover page with your personal information and answered all of the assignment questions, scan your assignment pages (including this cover sheet and in the correct order) and convert them into a single PDF submission. If you do not have a scanner, you may use a cellphone to take pictures of each page, then use a program like Adobe Acrobat Scan to create a single PDF. (Remember to use appropriate lighting, as dim or blurry pictures that cannot be read will not be marked.) Do not upload individual pages. Upload your single PDF submission to the **Dropbox labeled Assignment 3** on CourseLink.

(2) While in CourseLink, now click on the **Quizzes tab** on the navigation bar. On the Quiz List page, you will see a quiz called **Assignment 3**. Click Start Quiz. Then, answer each question using the final answers that you have already determined. As you complete each question, the answer will be automatically saved. You can see which questions have saved answers in the Questions section of the quiz's left panel. You can also click the question number in the quiz's left panel to go back to the question. Click **Go to Submit Quiz** after you answer all quiz questions.

Assignment 3 is due **11:59pm, October 2, 2020**. As stated in the course outline, assignments can be at most 2-days late without penalty. This corresponds to 11:59pm, October 4, 2020. No additional extension will be granted for any reason. Please direct questions to the teaching assistants during their office hours.

Best of success!

1. Name the computer programming language that (symbolically) represents the binary instructions of the processor. Write your answer below. [1 point]

Machine code

2. In an assembler, instruction addresses are counted by the [1 point]
   a. Program counter
   b. Location counter
   c. Line number
   d. None of the above

3. The assembler directive DS.W $10 [1 point]
   a. Reserves 10 words in memory
   b. Defines a constant to be $10
   c. Reserves 16 words in memory
   d. one of the above

4. In the space below, write a single 68000 assembler directive to initialize five consecutive longwords in memory. The longword values are 1,2,3,4, and 5, and should be specified (i.e., written) in binary. Label the first longword **start**. [2 points]

START DC.L %1, %10, %11, %100, %101

5. Write a single 68000 assembler directive to reserve memory for 100 words. [1 point]

DS 100

<u>Use the following information for questions 6-8:</u>

Assume that EQU directives have been used to assign the label **start** the value $0C04 and the label **last** $1000. Determine the value 16-bit hexadecimal value computed by the assembler for the following expressions:

6. start-6 [1 point] $0BFE

7. last – start [1 point] $03FC

8. (last – start)/2 [1 point] $01FE

9. Consider the two lines of code below. The last line seeks to use an assemble-time expression to compute the number of words in the buffer. The expression is wrong. Show the correct assemble-time expression in the space below. [1 point]

```
BUFFER  DS.L 30
LENGTH  DC.B (*-BUFFER)
```

LENGTH DC.B (*-BUFFER)/4

10. Write a single 68000 instruction to load the number of long words between the location given by label **start** (first location) and the label **last** (first location past the data) into the least-significant byte in data register D0. Hint: use an assemble-time expression for the source operand. [2 points]

MOVE.L #((last-start)/4), D0

11. Are the two following fragments of code equivalent? Explain. [2 points]

```
ORG     HERE
MOVE.B  #100,D0
MOVE.B  D0,Data
```

```
        ORG   HERE
Data    DC.B  100
```

No. The left code modifies a variable. The right code defines a constant.

12. In lecture, we said that due to their simplicity many assemblers do not allow *forward references*. Does the Easy68K allow forward references? Hint: Try it out! [1 point]

No.

13. Draw a memory map to illustrate the following code. Make sure to show all addresses and the contents of memory locations in hexadecimal, as well as the location of the labels. [3 points]

```
        ORG   $1000
BUF     DS.B  3
VAR1    DC.W  12
VAR2    DC.L  *
        DC.B  'AB'
```

BUF	$1000	
	$1001	
	$1002	
	$1003	
VAR1	$1004	$00
	$1005	$0C
VAR2	$1006	$00001006

Use the following information for questions 14-16:

Hand compile the following C global variable declarations using 68000 assembler directives. Assume that INTs are 32-bits, SHORTs are 16-bits, and CHARS are 8-bits.

14. char a[] = "Zoom"; [1 point]

a DC.B 'Zoom'

15. short b; [1 point]

b DS.W 1

16. int c = 0x123456; [1 point]

c DC.L $123456