Lab 7: Change Calculation Engine

Summary:
Practice assembly language programming by creating and testing a useful cash register change calculation program that will display results on an LCD display.

Learning Objectives:
- Review of subroutines and I/O ports
- Experience using an LCD display output device

Resources and Supplies:
- *LCD.asm* subroutine to drive the LCD display
- Text editor & 68HC12 assembler
- MON12 monitor program
- CML12S-DP256 development board
- CML12S-DP256 MON12 manual

All documents, including *LCD.asm* code, are available on the class website

Important Reminders:
- It is your responsibility to save the programs you create.
- Pre-lab assignments must be completed before coming to the lab.

Background:
INCLUDE Directive
It is often desirable to have blocks of code with specific, reusable, functions stored in individual files rather than pasting them as subroutines into the main program file. The *include* directive links an external .ASM program file into the main program, allowing it to be used as if it were a subroutine within the main program. To *include* a .ASM program file in your main program, use the following directive format:

```
#include "name.asm"
```

within the main loop (after ORG statement and before END) of your program using exactly this format (file name in quotes). The file you want to include (*name.asm*) must be stored in the same directory as your main program. The *include* statement should be placed in your program where you would have put the code contained within the included file. To call a subroutine from an *include* file, simple refer to the label at the top of the *include* file code.

LCD Display ASM Subroutine
The AXM-0295 project development board contains a liquid crystal display (LCD) that can be used to display output values. Control of the LCD is complex, so a subroutine has been written for you. The LCD subroutine is a file named *LCD.asm* and should be downloaded from the class website and stored in your ASM program directory.

You can access this subroutine using the *include* directive. For example, you can put the following line at the bottom of your main program, after the typical SWI instruction used to terminate a program under development:

```
#include "LCD.asm"      ; include lcd display routine, label = Display
```

You should put subroutines at the bottom of your main program, after the typical SWI instruction

used to terminate a program under development, but before the END directive.

Including the LCD.asm code within your program will permit you to call the *DISPLAY* subroutine as you would any other subroutine, i.e. JSR DISPLAY. The *DISPLAY* subroutine will setup the LCD device and send the value stored in $2000 for display on the LCD in binary format (8 characters on the LCD, each representing a binary bit of the value in $2000). Thus, whatever data you have stored in $2000 will show up on the LCD after calling the DISPLAY subroutine.

Programming Assignment

A company that designs microcontroller-based cash registers for retail markets has just hired you! A 68HC12 is to be used as the processing element of their new cash register that automatically distributes correct change. Your job is to develop a part of the cash register software that calculates the correct coins to be returned to a customer after a purchase. The fractional dollar amount (between $.00 and $.99) of the purchase, represented in hexadecimal, will be stored at a specified memory location ($1000). Your program must read this value from memory. Subtracting this value from $1.00 will determine the customer's change that is due, some amount between 0 and 99 cents. Your program must calculate the number of each type of coin (quarters, dimes, nickels, and pennies) to be returned to the customer. The output of your program will be used to control a coin release device for an automated change delivery system.

The maximum amount of change is $0.99. If we always use the largest value coins possible, the maximum number of coins is:

quarters, 3 (11 binary)
dimes, 2 (10 binary)
nickels, 1 (1 binary)
pennies, 4 (100 binary)

Adding up the binary bits indicates that a single 8-bit value, let's call it COIN, can indicate the desired number of each type of coin. As shown below, the 8-bit COIN vector consists of 2 bits for quarters (Q1-Q0), 2 for dimes (D1-D0), 1 for nickels (N0) and 3 for pennies (P2-P0).

bit	7	6	5	4	3	2	1	0
	Q1	Q0	D1	D0	N0	P2	P1	P0

Bit assignments for the COIN data vector.

Valid decimal and binary values for the COIN bits are:

Bits	Decimal	Binary
Q(1:0)	0 – 3	00, 01, 10, 11
D(1:0)	0 – 2	00, 01, 10
N(0)	0 – 1	0, 1
P(2:0)	0 – 4	000, 001, 010, 011, 100

The COIN vector should be calculated based on the fractional dollar amount of the customer's purchase price, which will be stored in $1000. After calculating the COIN vector, it should be saved to $2000 and then displayed on the LCD using the DISPLAY subroutine.

Example: If the fractional dollar purchase amount is $0.13, then the value stored in $1000 should be $0D_{16}$ (13_{10}). The program should calculate that 87 cents is to be returned to the customer. The program should determine that the 87 cents should be in the form of 3 quarters, 1 dime, 0 nickels and 2 pennies. Thus, the COIN vector should be set to 11010010, which would finally be displayed on the LCD.

The challenging parts of this program are:

1) Properly counting the number of each coin needed
2) Pasting together all of the count values into a single COIN vector byte. For this task, logic shift or bit set instructions should be considered.

Begin the main program at memory location $4000. The memory location for storing the fractional dollar purchase value is $1000. The value in memory location $2000 will be sent to the LCD by the DISPLAY subroutine. End the program with a software interrupt (SWI) instruction before including the subroutine.

Pre-lab Assignment:

- Read this entire lab assignment so you know what to expect in the lab.
- Complete the steps described in the Pre-lab sheet near the end of this document. Each student must complete his/her own pre-lab before coming to the lab and hand it in to the lab TA at the beginning of the lab.

Laboratory Assignment:

- Print the check off sheet. Where indicated, you must record your results on the check-off sheet. After you successfully finish the lab, show the TA your results and ask him to sign the check-off sheet.

Change Calculation Engine

Software Preparation:

1. Using ***WinIDE*** or any text editor, write the code for the change calculator program described in the Background section. Save it as a .ASM file in the program directory created previously to store the programs you develop for this lab. You will need to have the LCD.asm file (from class website) stored in this directory also.
2. Assemble your program file and fix any syntax errors.
3. If you wish, simulate your program and correct any bugs. You can do this with a simulator first or do all your debugging on the hardware.
4. Launch the ***AsmIDE*** utility from the 68HC12 program folder under the 'Start' menu.

   Within AsmIDE, select *File* → *Open* to open your .asm file.
5. From the *Build* menu, select *Assemble*. If the input code contains any errors, the *.lst* file will give the line number of the error and a brief description of the nature of the error.

Hardware Preparation:

6. From the PC on your lab bench, launch the ***AxIDE*** utility from the 68HC12 program folder under the 'Start' menu.
7. Connect the CML12S-DP256 board to your computer using the wiring bundle attached to your computer.
8. Connect power to the CML12S-DP256 board. Then press the **reset button**
9. In the AxIDE terminal window on the PC, **press ENTER** to start the Monitor program.

10. At the Monitor prompt (>), **type *'load'*** and **press ENTER**. Then upload the ***.s19*** record for your change calculation program.

Note: Refer to page 10 of the *CML12S-DP256 MON12 manual* for information on any MON12 command or type 'help' at the prompt for a list of commands. The most useful command are:

MD <address>	***Memory display***	***RD***	***CPU Register display***
MM <address>	***Memory modify***	***RM***	***CPU Register modify***
G	***Go = run program***	***T***	***Trace = step***
BR <address>	***Set Breakpoint***		

Testing:
Your program requires data at memory location $1000. To test the program you will need to store it manually.

11. Use the ***RM*** to set the Program Counter to point at address $4000.
12. Use the ***MM*** command to enter the appropriate hexadecimal input value for $0.13 in memory location $1000. Record this value on the check-off sheet.
13. Calculate the expected number of coins as change for a fractional dollar value of $0.13. Record these values and the resulting encoded COIN vector value.
14. Use the ***G*** command to start program execution.
15. Check your program results and debug your code until it is working correctly. Remember the ***T*** and ***BR*** commands if you need to track down bugs within your code. Remember to reset the PC to $4000 each time you re-run the program.
16. When the program is running correctly, record the result of the program (LCD display value) in the check-off sheet.
17. Record a comment of your observations and print your *.lst* file to include in your lab report.
18. Repeat Steps 11 - 16 for the fractional monetary input values below. Record the expected and observed program results (COIN value) on the check off sheet.
    a. $0.17
    b. $0.50
    c. $0.99
19. When you are satisfied that the program is operating correctly, ask the TA to check a demonstration of your program. Ask the TA to sign your lab check-off sheet.

Final Tasks and Notes

- Turn off the power supply and anything else that you might have turned on.
- Disconnect and return the microcontroller interface board to the lab closet.

Discussion Points

As explained in the *Lab Report Guide*, you should address these discussion points in a designated section of your report.

1. In the lab you used an *include* directive for a subroutine that is in another file. After looking at the .lst file, explain the purpose of the include directive in your own words.

.

2. Answer the questions below to develop a second program that calculates the number of dimes and pennies, only, to be returned to the customer; no quarters or nickels. The maximum amount of change is again $0.99.
    a. If we always use the largest value of coins possible, what is the maximum number of dimes returned? What is the maximum number of pennies returned?
    b. Write the new 8-bit COIN vector including the bit assignments.
    c. What are the valid decimal and binary values for the new COIN bits?

3. Assuming that $1 coins someday became popular in America and that pennies were dropped from our monetary system (rounding to the nearest $0.05 value), could a single 8-bit value still be used to represent the number of coins? Assume first that $1 coins would be used for all values less than $10. Would the answer change if $1 coins were only given as change for values less than $5?

PRE-LAB 7

Due: At the beginning of lab.

Student Name: _____ **Lab. Section (time):** _____

Make sure you read the lab document before you start the pre-lab, especially the Background section. For questions 1 and 2, attach the ASM code segments to this sheet.

1. Write an ASM code segment to calculate the number of quarters contained in a fractional dollar amount (between $0.01 and $0.99) that is represented by a hexadecimal value stored in memory location $1000. Store the output at memory location $2000. Start your code at memory location $4000. Try to use a simulator to test that your code works.

2. The maximum number of quarters in a fractional dollar amount is 3. Two bits are needed for the number of quarters. Write an ASM program segment that would store the number of quarters in memory location $2000 in such a way that bits 6-7 reflect the number of quarters and all other bits are zeros.

3. Assuming I have data in bits 4:3 of accA and data in bits 2:0 of accB, with all other bits irrelevant, how could I store the combined data into bits 4:0 of a single register? Describe the ASM instructions (or instruction sequence) that could be used.

4. What is the expected output (displayed on LCD) of the change calculation program (which only operates on the fractional dollar amount) for the following monetary input values?

   a. $3.17 → Displayed COIN value _____

   b. $5.50 → Displayed COIN value _____

   c. $1.99 → Displayed COIN value _____

LAB 7 CHECK-OFF SHEET

Student Name: _____ **Lab. Section (time):** _____

Complete this sheet as you complete the lab. Remember to have the TA check off each section of the assignment. This sheet must be included in your lab report.

For fractional dollar value = ***$0.13***

Step 12
Input hexadecimal value _____
Step 13
Number of Quarters: _____, Dimes: _____, Nickels: _____, Pennies: _____
Step 13
Encoded COIN vector value: Expected: _____
Step 16
Encoded COIN vector value: Observed: _____
Step 17: Comment on observations

_____

_____

For fractional dollar value = ***$0.17***

Step 12
Input hexadecimal value _____
Step 13
Number of Quarters: _____, Dimes: _____, Nickels: _____, Pennies: _____
Step 13
Encoded COIN vector value: Expected: _____
Step 16
Encoded COIN vector value: Observed: _____

For fractional dollar value = ***$0.50***

Step 12
Input hexadecimal value _____
Step 13
Encoded COIN vector value: Expected: _____
Step 16
Encoded COIN vector value: Observed: _____

For fractional dollar value = ***$0.99***

Step 12
Input hexadecimal value _____
Step 13
Encoded COIN vector value: Expected: _____
Step 16
Encoded COIN vector value: Observed: _____

TA sign off
Change Calculation Engine Initial_____