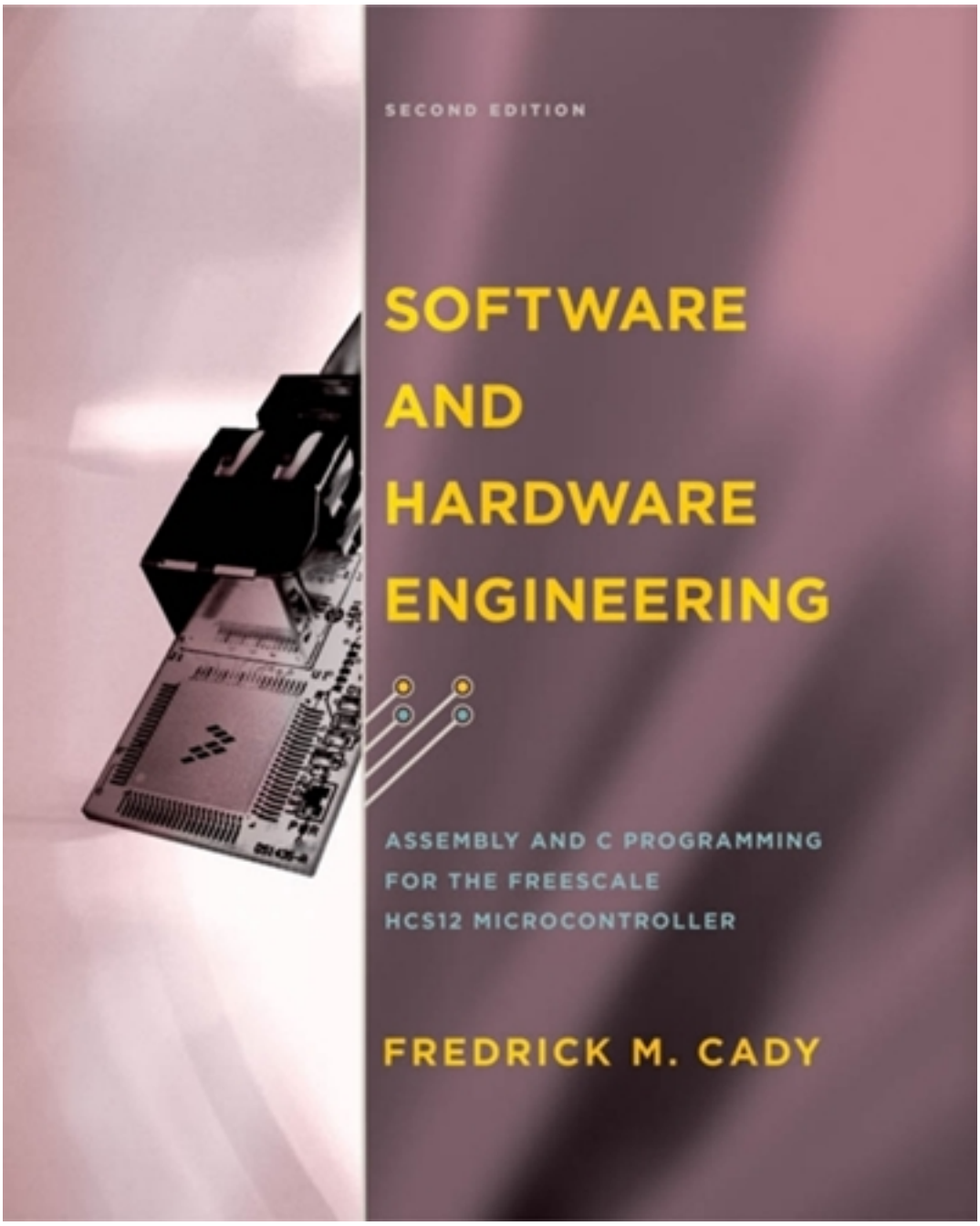
Solutions for Software and Hardware Engineering 2nd Edition by Cady

CLICK HERE TO ACCESS COMPLETE Solutions



Solutions

Software and Hardware Engineering
Assembly and C Programming
for the Freescale
HCS12 Microcontroller

Instructor's Solution Manual

Contents

Chapter Problems and Solutions

A letter or letters in […] at the end of each of the end of chapter problems signifies that the problem in some way tests that the student meets ABET accreditation criteria for Outcomes a – k as follows:

a. An ability to apply knowledge of mathematics, science, and engineering.
b. An ability to design and conduct experiments, as well as to analyze and interpret data.
c. An ability to design a system, component, or process to meet desired needs.
d. An ability to function on multi-disciplinary teams.
e. An ability to identify, formulate, and solve engineering problems.
f. An understanding of professional and ethical responsibility.
g. An ability to communicate effectively.
h. The broad education necessary to understand the impact of engineering solutions in a global and societal context.
i. A recognition of the need for, and an ability to engage in lifelong learning.
j. A knowledge of contemporary issues.
k. An ability to use the techniques, skills and modern engineering tools necessary for engineering practice.

CHAPTER 2 GENERAL PRINCIPLES OF MICROCONTROLLERS

Basic:

2.1 What is the difference between an assembler and a compiler? [a, c]

An assembler converts an assembly language program consisting of operations and their operands into the machine language (1s and 0s) for the microcontroller.

A compiler converts a high-level language, such as C, first into the assembly language needed for the line of C code, and then into the machine language for the microcontroller.

2.2 What is the advantage of a relocatable assembler compared to an absolute assembler? [a]

A relocatable assembler allows a program to be written in separate modules that are linked together to form the program. This permits modular design. An absolute assembler must have all its modules in one big program. This limits our ability to use pre-written and tested modules from a library of useful functions.

2.3 What is a microcontroller memory map? [a]

This shows what kind of memory is located in what address space.

2.4 What does a sequence controller do? [a]

Generates the sequence of control signals needed to execute an instruction.

2.5 Give short answers to the following: [a,g]

a. What is a data bus?

A parallel, bidirectional, binary information pathway with multiple sources and destinations.

b. Why is an address decoder used in I/O interfaces?

To select one-of-many sources or destinations.

c. How is an information source, such as a set of switches interfaced to a data bus?

With three-state gates whose enable is controlled by an Address_OK signal and a Read control signal.

d. What control signals are needed to latch data from the data bus into an output interface at the correct time?

Address_OK and Write

e. Give the sequence of events that occur when a CPU does an input (or read) cycle.

CPU puts address on the address bus
Address decoder generates ADR_OK
CPU asserts READ control signal
Input device puts data on the data bus
CPU reads the data
CPU de-asserts READ control signal

Intermediate:

2.6 Discuss the difference between an absolute and a relocatable assembler. [a, k]

Absolute Assembler: You must locate the code at assembly time and all code must be in one file. All defined labels are globally known.
Relocatable Assembler: A linker is used to locate the final code; modular software development with files separately developed and assembled may be used.

2.7 How do most microcomputer systems solve the problem of multiple sources of information present on a data bus? [g]

Multiple sources can exist as long as addressing and address decoding is used to enable one and only one source through three-state gates at any one time.

Advanced:

2.8 Why must a tristate gate be used to interface an input device to the data bus? [a, c]

The tristate gate allows multiple sources to source their data, one at a time, onto the data bus. The CPU's read control signal and an address decoder must control the tristate gate's enable.

2.9 Why must a latch be used to interface an output device to the data bus? [a, c]

The data bus is active all the time with data flowing to and from memory and I/O devices. To be able to output specific data to an output device at a specific time, a latch must be used which is clocked by the write control signal and the correct address.

2.10 For a CPU performing a write cycle, why does the CPU place the data on the data bus before asserting the WR_L control signal? [a]

To satisfy the data setup time requirement of the latch.

2.11 A microcontroller memory map shows 16K bytes of Flash EEPROM (ROM) in memory space $C000 - $FFFF and 1 Kbyte of RAM in memory space $1000 - $13FF. [c, k]

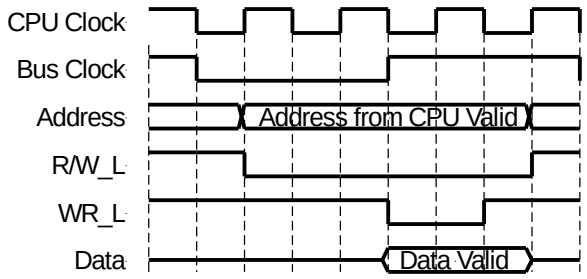
a. Give a range of addresses (in hex) suitable for locating code:

$C000 - $FFFF

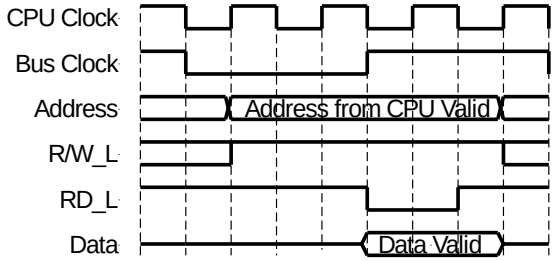
b. Give a range of addresses (in hex) suitable for allocating variable data storage:

$1000 - $13FF

2.12 Draw a timing diagram relative to the CPU clock shown in the figure below, which includes the address and data buses, R/W_L and the write control signal (WR_L = active low) and which shows a write cycle. [a]



2.13 Draw a timing diagram relative to the system CPU clock shown in the figure below, which includes the address and data buses, R/W_L, and the read control signal (RD_L = active low) and which shows a read cycle. [a]



CHAPTER 3 STRUCTURED PROGRAM DESIGN

Basic:

3.1. List at least five principles of top-down design. [a, c]

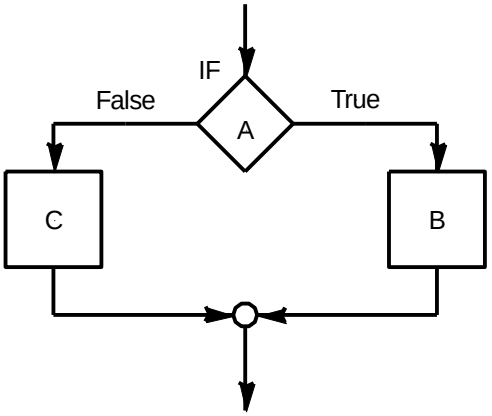
Understand the problem completely; design in levels; ensure correctness at each level; postpone details; successively refine your design; design without using a programming language.

3.2. What are the three basic elements of structured programming? [a]

SEQUENCE; IF-THEN-ELSE; REPETITION

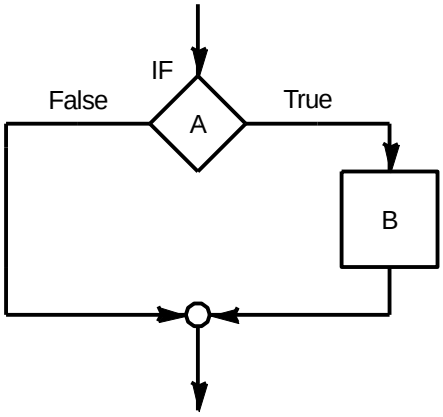
3.3. Write the pseudocode and draw the flow chart symbol to represent the decision IF A is TRUE THEN B ELSE C. [a, c]

```
IF A
THEN
   Begin B
   . . .
   End B
ELSE
   Begin C
   . . .
   End C
ENDIF A
```



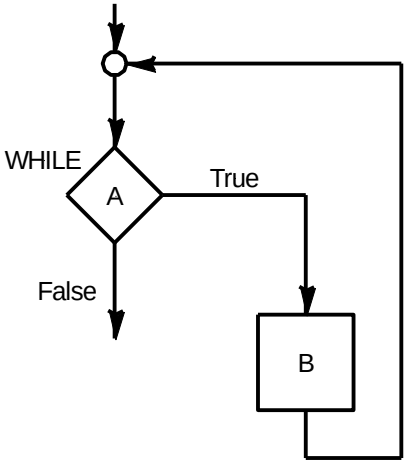
3.4. Write the pseudocode and draw the flow chart symbol to represent the decision IF A is TRUE THEN B. [a, c]

```
IF A
THEN
   Begin B
   . . .
   End B
ENDIF A
```



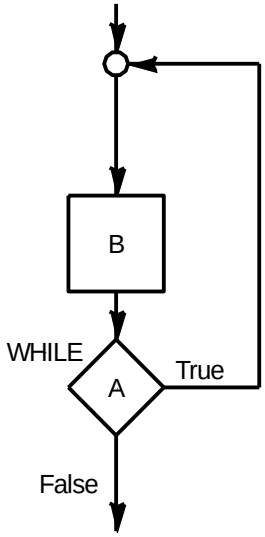
3.5. Write the pseudocode and draw the flow chart symbol to represent the repetition WHILE A is TRUE DO B. [a, c]

```
WHILE A
DO
   Begin B
   . . .
   End B
ENDO
ENDWHILE A DO
```



3.6. Write the pseudocode and draw the flow chart symbol to represent the repetition DO B WHILE A is TRUE. [a, c]

```
DO
   Begin B
   . . .
   End B
ENDO B
WHILE A
ENDO WHILE A
```



Intermediate:

3.7. Write a design using structured flow charts or pseudocode to implement the following problem description: [c]

Prompt for and input a character from a user at the keyboard.
If the character is alphabetic and is uppercase, change it to lowercase and output it to the screen.
If the character is alphabetic and is lowercase, change it to uppercase and output it to the screen.
If the character is numeric, output it with no change.
If it is any other character, beep the bell.
Repeat this process until an ESC character is typed by the user.

```
DO
   Output a prompt
   Input a character
   IF the character is alphabetic
   THEN
      IF the character is uppercase
      THEN
         Change the character to lowercase
      ELSE
         Change the character to uppercase
      ENDIF the character is uppercase
      Output the character to the screen
   ELSE
      IF the character is numeric
      THEN
         Output the character to the screen
      ELSE
         Output a bell to the screen
      ENDIF it is numeric
   ENDIF the character is alphabetic
ENDO
WHILE The character is not an ESC
```

Advanced:

3.8. Design a program that initializes an 8-bit data storage accumulator to 0 and then inputs 10_{10} successive 8-bit values from an input device located at address $70, adding each of them to the 8-bit data storage accumulator. If during this process an unsigned binary overflow occurs, print an error message and repeat from the beginning. Otherwise, after the 10 values have been input and added, output the result to an output device at location $71. Run the process forever. Your design must be a structured design and must show sequence, decision and repetition. [c]

```
DO
   Initialize accumulator to zero
   Initialize a counter to 10
   WHILE there is no unsigned overflow and counter > 0
   DO
      Get a value from $70
      Add it to the accumulator
      Save overflow indication
      Decrement counter
   ENDDO
   ENDWHILE there is no unsigned overflow and counter >0
   IF overflow occurred
   THEN
      Print the overflow error message
   ELSE
      Output the accumulator to $71
   ENDIF overflow occurred
ENDDO
FOREVER or WHILE(1)
```

3.9. Give a design using structured pseudocode to accomplish the following: [c]

A user is to input a character to select one of three processes. Valid characters are A, B, and C. A, B, and C select processes A, B, or C, respectively. Process A requires a byte of information to be input from an A/D converter, which it then converts to an integer decimal number in the range of 0 to 5 and displayed on the screen. Process B and C are not defined at this stage. Prompts and error messages are to be displayed. You do not have to give details of the decimal conversion required in Process A.

```
Prompt user for character A, B, or C
Get the character
IF the character is A
THEN
   Get input from the A/D
   Convert it to an integer decimal number in the range of 0 to 5
   Display the value on the screen
ELSE
   IF the character is B
   THEN
      Do process B
   ELSE
      IF the character is C
      THEN
         DO process C
      ELSE
         Print error message that user did not enter A, B, or C
      ENDIF the character is C
   ENDIF the character is B
ENDIF the character is A
```

3.10. Give a design using structured pseudocode to accomplish the following: [c]

A byte of data is to be input from an analog-to-digital converter and a critical value is to be input from a set of switches. If the A/D value is greater than the critical value, the microcontroller is to sound an alarm. Otherwise the alarm is to be turned off. This process is to continue forever.

```
DO
   Input data from A/D
   Input data from switches
   IF A/D value is greater than the switches value
   THEN
      Turn the alarm on
   ELSE
      Turn the alarm off
   ENDIF A/D value is greater than the switches
ENDDO
FOREVER or WHILE(1)
```

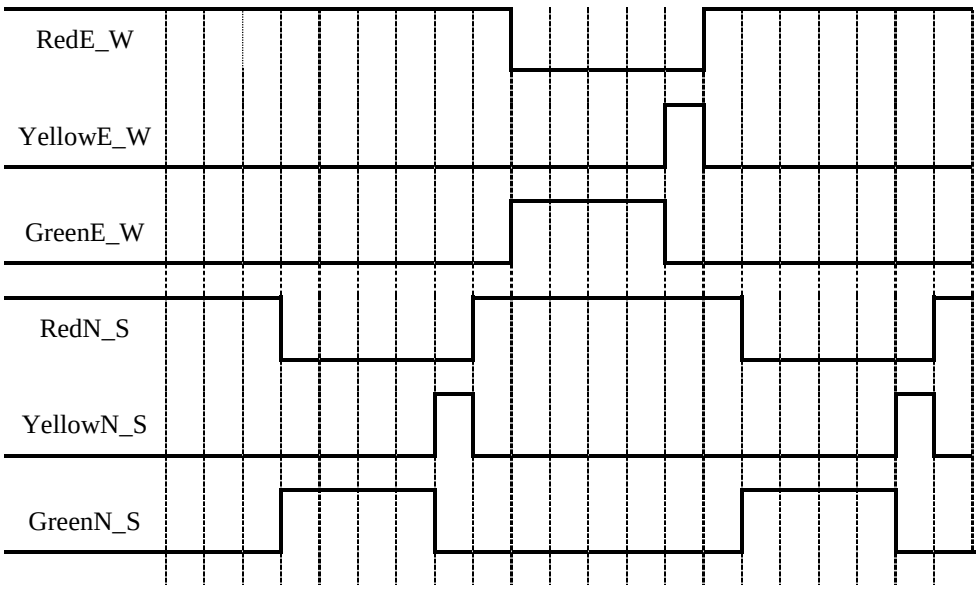
3.11. Design a traffic light controller: [c]

Imagine an intersection with North/South and East/West streets. There are to be six traffic light signals:

RedE_W, YellowE_W, GreenE_W

RedN_S, YellowN_S, GreenN_S

Assume the time elements in the table shown are 10 seconds and that a timer delay is available as a function or subroutine. Give the pseudocode structured design for the light controller.



```
Start with GreenN_S coming on:

Initial conditions: Turn RedE_W on and RedN_S on. All others off.
DO
   Turn RedN_S off and GreenN_S and RedE_W on.
   Wait 40 seconds
   Turn YellowN_S and RedE_W on and GreenN_S off
   Wait 10 seconds
   Turn RedN_S and RedE_W on and YellowN_S off.
   Wait 10 seconds
   Turn RedE_W off and GreenE_W and RedN_S on
   Wait 40 seconds
   Turn YellowE_W and RedN_S on and GreenE_W off
   Wait 10 seconds
   Turn RedE_W and RedN_S on and YellowE_W off
   Wait 10 seconds
FOREVER
```

CHAPTER 4 INTRODUCTION TO THE HCS12 HARDWARE

Basic:

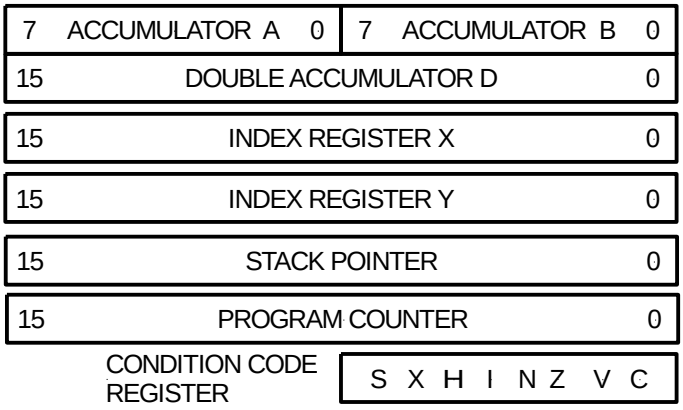
4.1. Which of the HCS12 ports is used for the A/D converter inputs? [a]

PORTAD

4.2. Which of the HCS12 ports is used with serial I/O? [a]

PTS for the SCI, PTM for the SPI

4.3. Draw the programmer's model for the HCS12. [a]



4.4. Which bits in the HCS12 condition code register may be tested with conditional branching instructions? [a]

N Z V C

4.5. Complete the following sentences to describe the operation of the bits in the condition code register: [a]

a. The N bit is set when *the most significant bit of the result is 1*.

b. The Z bit is set when *all bits of the result are 0.*

c. The V bit is set when *two's complement overflow occurs*.

d. The C bit is set when *carry out or borrow out occurs.*

4.6. Describe the following HCS12 addressing modes: [g]

immediate, direct, extended, indexed, indexed-indirect, inherent, relative.

Immediate: The data for the instruction immediately follows the op code. Immediate addressing is used for constants known at the time the program is assembled.

Direct: Direct addressing uses an 8-bit address to directly access a location in the first 256 bytes of memory.

Extended: Extended addressing uses a 16-bit address to access a memory location anywhere in the 64 Kbyte address space.

Indexed: Indexed addressing generates the effective address by adding a 5-, 9- or 16-bit signed constant offset (specified by the instruction) to the contents of the IX, IY, SP or PC registers. A 16-bit address is the result.

Indexed-Indirect: This mode uses indexed addressing to find the address of the memory location that contains the address of the data.

Inherent: Inherent addressing means the instruction itself specifies where the operand is located.

Relative: Relative addressing is used for branch instructions. An 8-bit, two's complement offset specified by the instruction is added to the contents of the program counter.

Intermediate:

4.7. Calculate the effective address for each of the following examples of indexed addressing. [a]

a. X = $0800
```
ldaa  0,X
```
EA = $0800

b. Y = $0800
```
staa  $10,Y
```
EA = $0810

c. X = $080D
```
ldaa  $25,X
```
EA = $0832

4.8. Discuss the relative advantages and disadvantages of direct and extended addressing. [g]

Direct addressing uses only eight bits to specify the address of the memory data and thus is faster and uses less memory than extended addressing. Its disadvantage is that only 256 memory locations can be addressed.

Extended addressing requires three (or more) bytes for the instruction and address and thus takes longer to execute and requires more memory. Extended addressing can access the whole 64 Kbyte address space.

4.9. Discuss the relative advantages and disadvantages of extended and indexed addressing. [g]

Indexed addressing is a two byte instruction (for the X register, three bytes for the Y register). Thus it is faster and uses less memory than extended addressing. Its major advantage is that the address of the data can be determined at run-time and the index register can be incremented and decremented to step through tables of data.

4.10. What is in the following CPU registers after a system reset? [a]

A, B, CCR, Stack Pointer.

Except for the I, S, and X bits in the CCR, the state of these registers after a system reset is undetermined.

Advanced:

4.11. What HCS12 addressing mode is best to use when you want to access a number of sequential elements in a data array? [a]

a. Immediate
b. Direct
c. Extended
d. Indexed
e. None of these

d. Indexed

4.12. What HCS12 addressing mode is best to use when you want to compare what is in accumulator A with a constant? [a]

a. Immediate
b. Direct
c. Extended
d. Indexed
e. None of these

a. Immediate

4.13. Discuss how the CPU fetches the first operation code of the first instruction to be executed following a system reset. [g]

After the system reset, the CPU fetches the address of the first op code to be executed from a special location in the ROM.

4.14. The HCS12 uses an instruction-fetch pipeline (or cache) that is filled by the CPU fetching bytes from memory while it (the CPU) is doing some other operations. For example, at the same time the CPU is adding the A to the B registers, which is an internal operation, it can be fetching another byte from memory to put into the instruction pipeline. [g]

a. Briefly explain how this strategy can improve the performance of the CPU.

The CPU must fetch the op code byte prior to executing the instruction. So if it can do it while executing another instruction, the total time/instruction can be reduced.

b. Briefly explain under what conditions this strategy might fail to improve CPU performance.

If the next instruction being executed is a branch to some other place in memory, the cache may not have the next op code.

CHAPTER 5 AN ASSEMBLER PROGRAM

Basic:

5.1 Give the assembler fields (label, opcode, operand, and comment) for the following instructions: [a]

a.
```
        staa    $800    ; Save the variable
        op code Operand  Comment
```

b
```
        bra     loop
        op code Operand
```

c.
```
loop:   ldaa    #$23    ; Initialize counter
Label   op code Operand  Comment
```

5.2 Give four ways to specify each of the following constants. [k]

a. The ASCII character X.

'X', "X", $58, 88, %01011000, @230

b. The ASCII character x.

'x', "x", $78, 120, %01111000, @270

c. 100_{10}

100, $64, %01100100, @144

d. 64_{16}

100, $64, %01100100, @144

5.3 Give the symbol used when specifying a constant in the following bases: [k]

hexadecimal, decimal, binary, ASCII

hexadecimal ($), decimal (None, decimal is the default base), binary (%), ASCII (' ')

5.4 Write a statement that allocates memory for a one-byte variable called COUNT. [k]

```
COUNT:  DS.B    1
```

5.5 Write a statement that creates a string constant in ROM named MSG that contains the text 'This is a message.'. [k]

```
MSG:    DC.B    'This is a message.'
```

5.6 Write a statement that allocates a byte constant COUNT in ROM and initializes it with the value $26. [k]

```
COUNT:  DC.B    $26
```

5.7 Write a statement that allocates a word-size constant BIGCOUNT in ROM and initializes it with the value $1234. [k]

```
BIGCOUNT:  DC.W    $1234
```

5.8 Show how to allocate storage for ten (decimal) bytes. [k]

```
data:   ds.b    10
```

Intermediate:

5.9 In the CodeWarrior assembler, describe the difference between DC.B and DS.B. [k]

DC.B defines a byte constant, allocates storage for it and initializes it in memory.

DS.B allocates a byte storage location without initialization.

5.10 Consider the term "Allocate data storage" [g]

a. What is meant by this term?

Reserve a memory location for the storage of variable data in the program.

b. What are the two lines in an assembly language program to allocate storage for a variable?

By locating the storage allocation in the correct memory, e.g.

DataLoc: SECTION

and then by reserving sufficient bytes of memory for the variable(s) using the define storage directive, e.g.

Count: DS.B 1

c. Why are variable data storage locations initialized at run time instead of at assembly time?

Two reasons. First the variable may need to be initialized with a value depending on the program, but mostly, because variable are stored in RAM and even though the variable initialization value may be known at assembly time, it must be initialized at run time because RAM does not retain its contents when the power is off.

5.11 What is an assembler expression ? [k]

A combination of symbols, constants, and algebraic operators. The assembler evaluates the expression to produce a value for the operand.

5.12 Give an example of an assembler expression. [k]

```
ldaa    (BUF2-BUF1),x
```

Advanced:

5.13 Give the result in hexadecimal for each of the following assembler expressions: [k]

```
MS_MASK:    EQU   %11110000
LS_MASK:    EQU   %00001111
MOST:       EQU   255
NEG:        EQU   -128
SIX:        EQU   6
TEN:        EQU   10
DATA1:      DS.B  10
DATA2:      DS.B  10
```

a. `Q1:  DC.B  SIX+TEN`

Q1 = $10

b. `Q2:  DC.B  DATA2-DATA1`

Q2 = $0A

c. `Q3:  DC.B  SIX | TEN`

Q3 = $0E

d. `Q4:  DC.B  MOST ^ LS_MASK`

Q4 = $F0

e. `Q5:  DC.B  ~MS_MASK`

Q4 = $0F

5.14 An assembly language programmer wants to multiply a variable data value stored in DATA1 by a constant 5. What is wrong with the following code segment? [k]

```
ldaa   DATA1*5
staa   DATA1
```

The '' is an assembler expression operator and is used to evaluate constants at assembly time. It does not produce code to be used at run time.*

CHAPTER 6 THE LINKER

Intermediate:

6.1 Answer the following questions based on the given linker parameter file: [k]

a. How much RAM is in this microcontroller?

$0800 = 2048 bytes

b. Where is the RAM located?

$0800 - $0FFF

c. Where is the program memory located?

$C000 - $FEFF

d. How many bytes have been allocated for the stack?

$0100 = 256 bytes

```
SEGMENTS
    RAM = READ_WRITE 0x0800 TO 0x0FFF;
    /* unbanked FLASH ROM */
    ROM_4000 = READ_ONLY  0x4000 TO 0x7FFF;
    ROM_C000 = READ_ONLY  0xC000 TO 0xFEFF;
    /* banked FLASH ROM */
END
PLACEMENT
    DEFAULT_ROM                 INTO  ROM_C000;
    DEFAULT_RAM                 INTO  RAM;
END
STACKSIZE 0x100
VECTOR 0 Entry  /* reset vector: this is the default entry point for a
Assembly application. */
VECTOR ADDRESS 0xFFDE tof_pa0_isr
```

6.2 Answer the following questions based on the given linker parameter file: [k]

a. How much RAM is in this microcontroller?

$1800 = 6144 (6K) bytes

b. Where is the RAM located?

$0800 - $1FFF

c. Where will MySubData be located?

In the RAM.

d. Where will MyData be located?

In the RAM.

e. Where will MainConst be located?

In the ROM_C000

```
SECTIONS
    RAM = READ_WRITE 0x0800 TO 0x1FFF;
/* unbanked FLASH ROM */
    ROM_C000 = READ_ONLY  0xC000 TO 0xFEFF;
END
PLACEMENT
/* Place the code and constant sections in ROM */
    .init, MainCode, SubCode,MainConst, SubConst,
    DEFAULT_ROM                                    INTO  ROM_C000;
/* Place the Variable Data */
    MyData, MySubData,DEFAULT_RAM                  INTO  RAM;
END
/* Define the stack size needed */
STACKSIZE 0x100
/* Specify the initial reset vector entry point */
VECTOR 0 Entry  /* reset vector */
/* INIT is needed for assembly applications */
INIT Entry
```

Advanced:

6.3 Inspect the given (partial) map file and answer the following questions: [k]

a. What source files are used in this program?

rel_ex_1a_main.asm, rel_ex_1a_sub_1.asm

b. What is the program entry point?

$C000

c. How many bytes of memory are allocated in the RAM?

$102 = 258 bytes

d. To what memory location does the label "loop" refer to?

$C009

e. In what memory location will you find the main_data variable?

$0800

f. In what memory location will you find the sub_data variable?

$0801