Homework 5 – Solutions

Chapter 6

6.1

Yes, for example, an iterative block where the test condition remains true for each iteration. This procedure will never end and is therefore not _nite and not an algorithm. The following is an example of a procedure that isn't an algorithm:

x3000 0101 000 000 1 00000 (LOOP AND R0, R0, #0)
x3001 0000 010 111111110 (BRz LOOP)

This is not an algorithm because the branch instruction is always taken and the program loops inde_nitely.

6.2

Problem Statement: Subtract the value in R1 from the value in R2 and store the result in R0. The following is the systematic decomposition of the problem concluding with an LC-3 program solving the problem.

(a) Start -> Subtract R1 from R2; place result in R0
-> End

(b) Start -> Compute the complement of the value in R1
-> Add this result to R2 and store the result in R0
-> End

(c) Start -> Negate R1 -> Add 1 to R1 ->
Add R1 and R2; store result in R0 -> End

(d) 1001 001 001 111111 (NOT R1, R1)
0001 001 001 1 00001 (ADD R1, R1, #1)
0001 000 001 0 00 010 (ADD R0, R1, R2)

6.15

0111 010 100 000111 (STR R2, R4, #7)

6.16

x3000 0010 000 0 0000 0100 (LD R0, #4)
x3002 1011 000 0 0000 0011 (STI R0, #3)
x3003 1111 0000 0010 0001 (TRAP x21)
x3006 0100 0000 0000 0001

Chapter 7

7.7

The assembly language program is:

```
.ORIG x3000
AND R5, R5, #0
ADD R5, R5, #1 ;R5 will act as a mask to
;mask out the unneeded bit
AND R1, R1, #0 ;zero out the result register
AND R2, R2, #0 ;R2 will act as a counter
LD R3, NegSixt
MskLoop AND R4, R0, R5 ;mask off the bit
BRz NotOne ;if bit is zero then don't
;increment the result
```

```
ADD R1, R1, #1 ;if bit is one increment
;the result
NotOne ADD R5, R5, R5 ;shift the mask one bit left
ADD R2, R2, #1 ;increment counter (tells us
;where we are in bit pattern)
ADD R6, R2, R3
BRn MskLoop ;not done yet go back and
;check other bits
HALT
NegSixt .FILL #-16
.END
```

7.10

Add R3, R3, #30 contains an immediate value that is too large to be stored in the Add instruction's immediate value. This instruction cannot be translated by the assembler, thus the error is detected when the program is assembled, not run on the LC-3.

7.13

Error 1:
Line 8: ST R1, SUM
SUM is an unde_ned label. This error will be detected at assembly time.
Error 2:
Line 3: ADD R1, R1, R0
R1 was not initialized before it was used; therefore, the result of this ADD instruction may not be correct. This error will be detected at run time.

7.17

There is not a problem in using the same label in separate modules assuming the programmer expected the label to refer to different addresses, one within each module. This is not a problem because each module has its own symbol table associated with it. It is an error on the otherhand if the programmer expected each label AGAIN to refer to the same address.

7.18

a) LDR R3,R1,#0
b) NOT R3,R3
c) ADD R3,R3,#1
or
a) LDR R3,R1,#0
b) NOT R4,R4
c) ADD R4,R4,#1

7.24

When the BR LOOP instruction is executed, it checks the value of the condition codes, which have been set based on the value written into R3 as a result of the ADD R3, R3, R3 instruction. The condition codes are not changed by the branch instruction, so when the branch back to the label LOOP is taken, the next branch instruction (BRz DONE) will also use the condition codes as set by the value written into R3. However, the BRz DONE instruction should be branching based on the value in the register that is used

to keep track of the loop counter, which is R2. This problem can be fixed by switching the instructions ADD R2, R2, #-1 and ADD R3, R3, R3.

Chapter 8

8.1

(a) A device register is a register (or memory location) that is used for data transfer to/from an input/output device. It provides a means of communication between the processor and the input/output device. The processor can poll this register to _nd out whether it has received an input or it can send an output from/to the speci_c device that the device register belongs to. In memory mapped I/O device registers are dedicated memory locations for each I/O device. There may be more than one device register (dedicated memory location) for one device.

(b) A device data register is a device register (a dedicated memory location in memorymapped I/O) that holds the data that is to be input/output.

(c) A device status register is a device register (a dedicated memory location in memorymapped I/O) that indicates the status of the input/output. It allows for the processor to know whether or not input/output of the value in the device data register has occurred. Basically it is an important step to achieve synchronization in an asynchronous I/O system.

8.2

A ready bit is not needed if synchronous I/O is used because the processor will know exactly when the data will arrive and when it will be taken away (input and output). It will do input and/or output at regular intervals, and it will be guaranteed that during those intervals the input data is taken by the computer and the output data goes to the output device.

8.4

(a) The interaction between a remote control and a television is synchronous. The television samples at speci_c intervals to see if a key on the remote control has been pressed. No synchronization is needed in this transaction.

(b) The interaction between the mail delivery person and you is asynchronous. Neither do you check your mail at regular intervals, nor does the mail delivery person come at the same time everyday. Instead you use the mailbox as a synchronization mechanism (much like the .Ready bit.). Some mailboxes are located at the street, rather than at the door. They usually come equipped with a _ag that the mail delivery person lifts when depositing mail, and you lower when removing mail. The _ag is very much a ready bit.

(c) The interaction between a mouse and the PC is synchronous. The PC samples mouse movements at speci_c intervals. At each interval, the direction and speed of the mouse is read by the PC. No synchronization is needed in this interaction.

8.7

Memory mapped and polling. The system is memory-mapped because KBSR and KBDR device registers have assigned addresses in the memory address space of the ISA. The system is polling because the Ready bit is tested to see if a key has been struck.

8.11

Interrupt-driven I/O is more ef_cient than polling. Because, in polling, the processor needs to check a speci_c register (or memory location) regularly to see if anything is being input

or output. This consumes unnecessary processing power because the processor checks the register periodically (stopping all other jobs) even when nothing is being input or output. (Most of the time the register will not be inputting or outputting anything unless it is a really I/O-intensive program). However, in interrupt-driven I/O, when something is input or output by a device, the device sends a signal to the processor. Only when the processor receives that signal, it stops all other jobs and does the I/O. Hence, processing power is used for I/O only when it is necessary to do so.

8.13

Suppose the LC-3 datapath allows combining the two registers into one. Using separate registers, the test to see if the Ready bit is set simply involves checking bit 15 of the status register. This is performed using a branch instruction that tests if the value of the register is negative. If the KBSR and DSR are combined, the test to see if the Display device Ready bit is set involves masking out bit 14 and testing if the bit is set or cleared. Doing it this way requires more instructions than the first method.

8.15

NOTE: Please refer to the errata for the new problem statement.

(a) The keyboard interrupt is enabled, and the digit 2 is repeatedly written to the screen.

(b) The character typed is echoed twice to the screen.

(c) The digit 2 some number of times, followed by the digit typed twice or three times, followed by the digit 2 continually thereafter.

(d) The digit typed will be displayed to the screen twice or three times, depending on when the typed character interrupted the program. If the program was interrupted immediately after LD R0, B , the character typed would appear on the screen three times.