# ECE 2560 Quiz 4

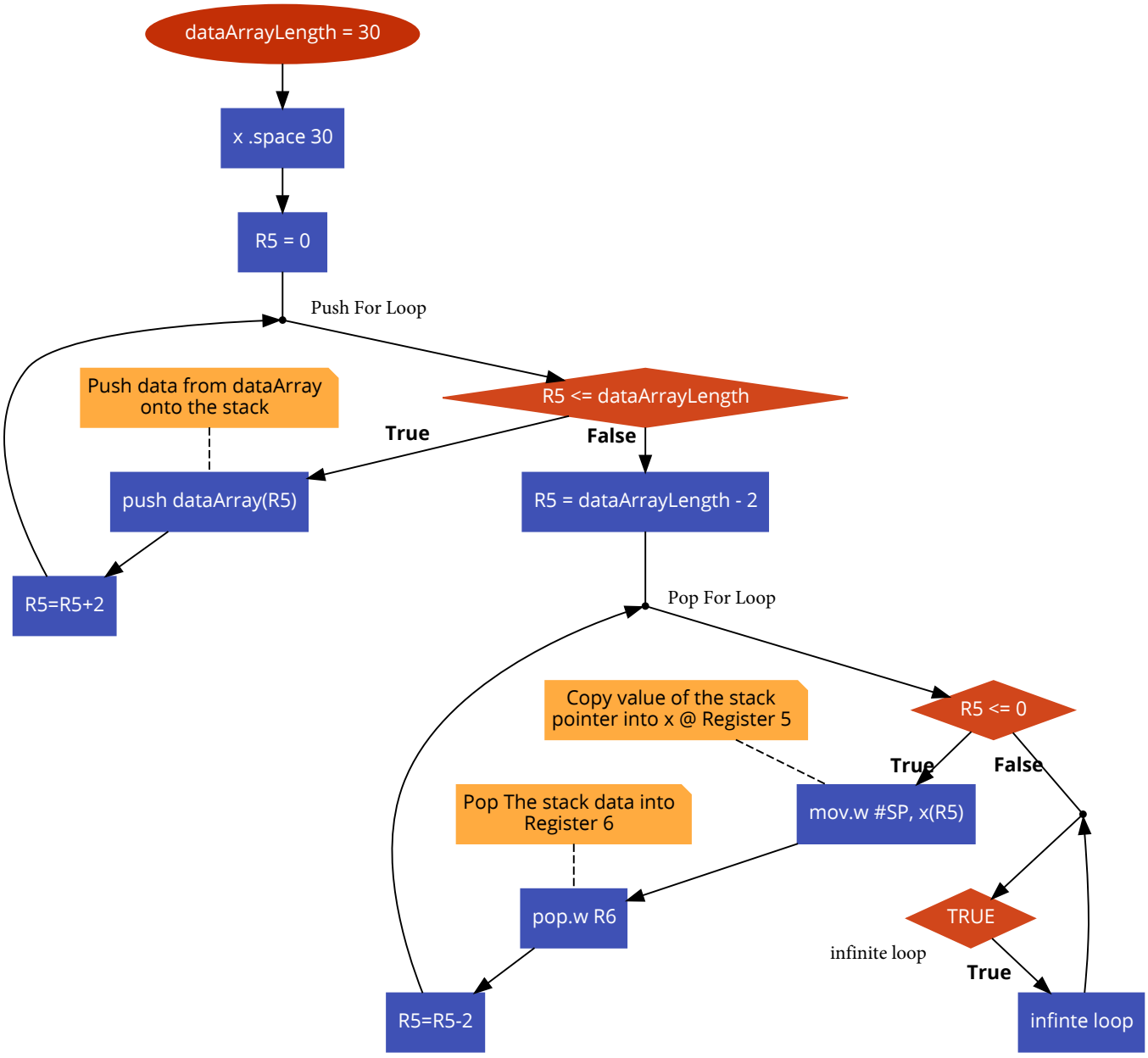
Matt Gambill
gambill.46@osu.edu

March 23, 2019

## Question 1:

```
1
2  dataArrayLength = 30;
3  x: .space 30 // Inialize 12 words of RAM
4
5  for ( R5 = 0; R5 <= dataArrayLength; R5=R5+2) {
6    push dataArray(R5); // Push data from dataArray onto the stack
7  }
8
9  for(R5 = dataArrayLength - 2; R5 <= 0; R5=R5-2){
10   mov.w #SP, x(R5); // Copy value of the stack pointer into x @ Register 5
11   pop.w R6;  // Pop The stack data into Register 6
12 }
```

**Question 2:**


dataArrayLength = 30
x .space 30
R5 = 0
Push For Loop
Push data from dataArray onto the stack
R5 <= dataArrayLength
True
False
push dataArray(R5)
R5 = dataArrayLength - 2
R5=R5+2
Pop For Loop
Copy value of the stack pointer into x @ Register 5
R5 <= 0
True
False
Pop The stack data into Register 6
mov.w #SP, x(R5)
pop.w R6
TRUE
infinite loop
True
R5=R5-2
infinte loop

## Question 3:

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include device header file

;-------------------------------------------------------------------------------
            .def    RESET                   ; Export program entry-point to
                                            ; make it known to linker.
;-------------------------------------------------------------------------------

      .data ; data region
      .retain
      .retainrefs
x:      .space 30
;-------------------------------------------------------------------------------

            .text                           ; Assemble into program memory.
            .retain                         ; Override ELF conditional linking
                                            ; and retain current section.
            .retainrefs                     ; And retain any sections that have
                                            ; references to current section.

;------------------------ Fixed Data in ROM Creation ---------------------------

dataArray:  .word 0x1111, 0x2222, 0x3333, 0x4444, 0x5555, 0x6666, 0x7777
      .word 0x8888, 0x9999, 0xAAAA, 0xBBBB, 0xCCCC, 0xDDDD, 0xEEEE
      .word 0xFFFF

dataArrayLength:  .word 0x001e
zero: .word 0x000
two:  .word 0x002
;-------------------------------------------------------------------------------
RESET       mov.w   #__STACK_END,SP         ; Initialize stackpointer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer


;-------------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------------

        mov.w   &zero, R5       ;  Move Zero into R5

push_for_loop:
        cmp.w   &dataArrayLength, R5  ;  Comapre  R5 to the number 8
        jge    exit_push_for_loop     ;  Exit For loop

        push.w  dataArray(R5)      ;  Push Data onto the Stack
        add.w   &two, R5        ;  Add 2 to R5
        jmp    push_for_loop     ;  Restart Loop
exit_push_for_loop:

        sub.w   &two, R5        ;  Adjust R5 to (dataArrayLength - 2) to not include
    __STACK_END in x
pop_for_loop:
        cmp.w   &zero, R5       ;  Compare R5 to the number 0
        jge    istrue           ;  Jump to is true if R5 >= 0
        jmp    exit_pop_for_loop    ;  Exit Loop if R5 <= 0

istrue:
        mov.w   SP, x(R5)       ;  Move SP onto Heap (x)
        pop.w   R6              ;  Pop Stack Data into R6

        sub.w   &two, R5        ;  Decrement R5 by 2
        jmp    pop_for_loop      ;  Restart Loop
```

```
66 exit_pop_for_loop:
67
68
69 loop:       jmp    loop;
70
71 ;-------------------------------------------------------------------------------
72 ; Stack Pointer definition
73 ;-------------------------------------------------------------------------------
74             .global __STACK_END
75             .sect   .stack
76
77 ;-------------------------------------------------------------------------------
78 ; Interrupt Vectors
79 ;-------------------------------------------------------------------------------
80             .sect   ".reset"                ; MSP430 RESET Vector
81             .short  RESET
```

Registers

| Name | Value |
|---|---|
| Core Registers | |
| PC | 0xC072 |
| SP | 0x0400 |
| SR | 0x0005 |
| V | 0 |
| SCG1 | 0 |
| SCG0 | 0 |
| OSCOFF | 0 |
| CPUOFF | 0 |
| GIE | 0 |
| N | 1 |
| Z | 0 |
| C | 1 |
| R3 | 0x0000 |
| R4 | 0x0FF7 |
| R5 | 0xFFFE |
| R6 | 0x1111 |
| R7 | 0xFCF7 |
| R8 | 0xFEEB |
| R9 | 0x4B6F |
| R10 | 0x0000 |
| R11 | 0xB7FB |
| R12 | 0x545F |
| R13 | 0xF78E |
| R14 | 0xFBFF |
| R15 | 0x8FFF |
| Special Function | |

0x200 - x <Memory Rendering 1>

16-Bit Hex - TI Style

```
0x0200  .bss, .data, _bss, _data, _end, end, x
0x0200  03FE 03FC 03FA 03F8 03F6 03F4 03F2 03F0 03EE 03EC 03EA 03E8 03E6 03E4 03E2
0x021E  _edata, edata
0x021E  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x0250  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x0282  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x02B4  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x02E6  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x0318  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x034A  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x037C  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x03AE  0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x03E0  0000 FFFF EEEE DDDD CCCC BBBB AAAA 9999 8888 7777 6666 5555 4444 3333 2222 1111
0x0400  _STACK_END
0x0400  3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF
```