# NCSU ECE 306 - Summer 2003 - Midterm Solution

**1-25 worth 4 points each**

1. e
2. c
3. a
4. d
5. e
6. e
7. e
8. d
9. a
10. d
11. b
12. c
13. a
14. All are correct.
15. e
16. c
17. a
18. d
19. c
20. d
21. d
22. d
23. a
24. b
25. e

**26. 5 pts each line (or equivalent)**

```
LOOP:  dec.w -6[FB]
       cmp.w #0001h, -6[FB]
       jge LOOP
```

**27. 5 pts**

$$\frac{1500\ \text{mAhr}}{4\ \text{mA}} \cdot \frac{1\ \text{day}}{24\ \text{hr}} = 15.625\ \text{days}$$

**28. 5 pts** The dynamic link is FB, which points to the activation record. C identifiers are mapped to specific entries in the activation record by the symbol table at compile time.

**29. 5 pts** The flash memory inside the processor cannot be programmed while in execution mode. Since program code is stored in flash, it cannot be changed while a program is running.

**30. 5 pts** Microcontrollers have the abilty to be reprogrammed, whereas an ASIC cannot be changed after being manufactured. ASICs are faster than microcontrollers. Development cost for an ASIC is high, but the per-piece price is typically lower than microcontrollers.

**31. 5 pts** C allows more control of memory and of physical hardware. It is efficient: space- and time-wise. C is easier to program than assembly language.

**32. 2 pts each + 2 for trying** B, D, A, C

**33. a. 15 pts, 3 pts each term**

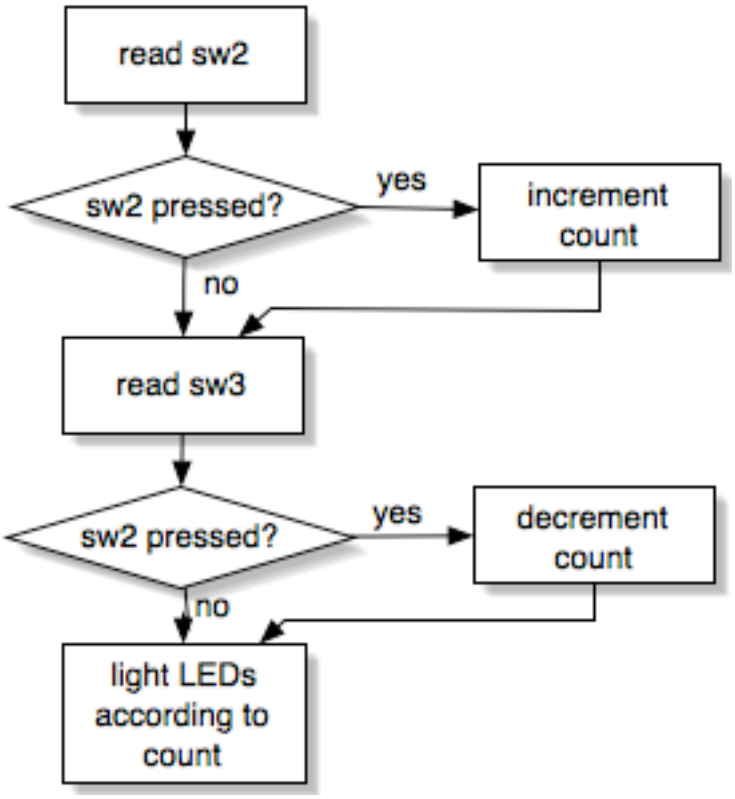
4 (args) + old FB (2) + return address (3) + auto vars (8) + pushed regs (6) = 23 bytes

**b. 15 pts, 5 pts each term, 5 pts for trying**

old FB (2) + auto vars (14) = 16 bytes

**34.**

**a. 15 pts - 5 pts sw2, 5 pts sw3, 5 pts lighting**




**b. 50 pts**
**10 pts - reading switches and changing variable . Must accept simultaneous button presses**
**10 pts - lighting LEDs**
**5 pts - comments**
**(others mentioned in code below)**

```
// main.c   J.Conrad  6/25/03
// Small program to count switch presses using a variable and
// the LEDs.  SW2, when pressed, will cause a variable to count
// up one for every press.  SW3, when pressed, will cause a
// variable to count down one for every press.

#include "sfr262.h"  // -2 pts for forgetting, -1 for using sfr10.h

#define LED_R p7_0  // all symbols should have been defined (5 pts)
#define LED_Y p7_1
#define LED_G p7_2

#define SW2 p10_5
#define SW3 p10_6
```

```
#define LED_ON (0)
#define LED_OFF (1)

unsigned int var_up_down; // Global var- count SW2/SW3 presses (5pts)

// ISR switch_isr   J.Conrad  6/25/03
// Works if both SW2 and SW3 pressed at same time
// Global variable var_up_down changed depending on SW2/SW3
#pragma INTERRUPT switch_isr  // 3 pts
void switch_isr () {   // 2 pts for correct function signature
  if (!SW2) var_up_down++;  // SW2 pressed
  if (!SW3) var_up_down--;  // SW3 pressed
  // Light LEDs (alternative below)
  if (var_up_down & 1) LED_G = LED_ON; else LED_G = LED_OFF;
  if (var_up_down & 2) LED_Y = LED_ON; else LED_Y = LED_OFF;
  if (var_up_down & 4) LED_R = LED_ON; else LED_R = LED_OFF;
}

// Main function - initialize ports and busy-wait loop
void main () {
  pd10_5 = pd10_6 = 0;    // Switches set as inputs - 5 pts
  pd7_2 = pd7_1 = pd7_0 = 1;       // LEDs set as outputs - 5 pts
  LED_R = LED_Y = LED_G = LED_OFF; //LEDs off
  var_up_down = 0;
  while (1);     // 5 pts for infinite wait loop
}

  // Alternate to ifs in ISR is this switch/case below
  //switch(switch_isr %8) {
  //  case (0): LED_R = LED_Y = LED_G = LED_OFF; break;
  //  case (1): LED_R = LED_Y = LED_OFF; LED_G = LED_ON; break;
  //  case (2): LED_R = LED_G = LED_OFF; LED_Y = LED_ON; break;
  //  case (3): LED_R = LED_OFF; LED_Y = LED_G = LED_ON; break;
  //  case (4): LED_R = LED_ON; LED_Y = LED_G = LED_OFF; break;
  //  case (5): LED_R = LED_G = LED_ON; LED_Y = LED_OFF; break;
  //  case (6): LED_R = LED_Y = LED_ON; LED_G = LED_OFF; break;
  //  case (7): LED_R = LED_Y = LED_G = LED_ON; break;
  //  default: break;  // not really needed
  //}
```

**c. 5 pts.** Need to modify correct vector in `sect30.inc`.
Change `.lword _dummy_int` of switch's entry to :

```
.glb _dummy_int
.lword _dummy_int
```