Problem 1.

Given the following lines of code, what are the contents at the beginning of RAM and Flash? Fill in the two tables below with known data.

```
.data
.retain
.retainrefs

course: .set 2560 ; Hex 0xA00
calendar:
  year: .space 2
  day: .space 1
  mon: .space 1
  time: .space 2

.text ; Assemble into program memory.
.retain
.retainrefs

initYr: .word 2016 ; Hex 0x7E0
initDM: .byte 15, 02 ; Hex 0xF, 0x2
initT: .byte 17, 20 ; Hex 0x11, 0x14
ram: .set 0x200
```

<table>
<thead>
<tr>
<th>Address</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x200</td>
<td>0x00</td>
</tr>
<tr>
<td>0x202</td>
<td>0x00</td>
</tr>
<tr>
<td>0x204</td>
<td>0x00</td>
</tr>
<tr>
<td>0x206</td>
<td>0x00</td>
</tr>
<tr>
<td>0x208</td>
<td>0x00</td>
</tr>
<tr>
<td>0x20A</td>
<td>0x00</td>
</tr>
</tbody>
</table>

Leave blank.
<table>
<thead>
<tr>
<th>Address</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xC000</td>
<td>0x07E0</td>
</tr>
<tr>
<td>0xC002</td>
<td>0x020F</td>
</tr>
<tr>
<td>0xC004</td>
<td>0x1411</td>
</tr>
<tr>
<td>0xC006</td>
<td>0x0000</td>
</tr>
<tr>
<td>0xC008</td>
<td>0x0000</td>
</tr>
<tr>
<td>0xC00A</td>
<td>0x0000</td>
</tr>
</tbody>
</table>

Problem 2: Given the code snippets below, what is the initial value of PC?

```
.OxC006

.text
.retain
.retainrefs

.initYr: .word 2016 ; Hex 0x7E0
.initDM: .byte 15, 02 ; Hex 0xF, 0x2
.initT: .byte 17, 20 ; Hex 0x11, 0x14

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

.reset
".reset" ; MSP430 RESET Vector
.short RESET
```

STOPWDT mov.w #WDTCTL, &WDTCTL ; Stop watchdog timer

; Stack Pointer definition

; ; .sect .stack

; ; Interrupt Vectors

; ; .sect ".reset" ; MSP430 RESET Vector
The code in problem 1 and 2 continued as following. Problem 3 – 5 are based on the code below:

```assembly
RESET       mov.w   __STACK_END,SP      ; Initialize stackpointer
StopWDT     mov.w   &WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer

; Main loop here

  mov.w   &initYr, &year
  mov.w   &initDM, &day
  mov.w   &initT, &time
  add.b   #55, &time+1
  mov.w   &ram, R15

loop1:  
  dec.w   R15
  jnz     loop1
  mov.w   #1, R15
  mov.w   #3, R14

loop2:  
  add.b   #1, calendar(R15)
  inc.w   R15
  dec.w   R14
  jnz     loop2

endLoop:  jmp     endLoop
```

Problem 3: What is the value of R15 when Program Counter (PC) reaches loop1?

2016 (Ox07E0)

Problem 4: What are the contents at the beginning of RAM and Flash when the program reaches endLoop? Fill in the two tables below with known data.

<table>
<thead>
<tr>
<th>Address</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x200</td>
<td>08</td>
</tr>
<tr>
<td>0x202</td>
<td>03</td>
</tr>
<tr>
<td>0x204</td>
<td>4B</td>
</tr>
<tr>
<td>0x206</td>
<td></td>
</tr>
<tr>
<td>0x208</td>
<td></td>
</tr>
<tr>
<td>0x20A</td>
<td></td>
</tr>
</tbody>
</table>

The address of time is Ox204, &time+1 means the value at address time+1=Ox205, which is Ox14. 55 in decimal is Ox37. Ox14 + Ox37 = Ox4B. When the program reaches loop2, R15=1, R14=3. loop2 is implemented 3 times, calendar(R15) is index mode, meaning the value at address calendar+R15, the address of calendar is Ox200. When loop2 is implemented first time, the value at address Ox200+1=Ox201 should add by 1, so Ox07 + 1 = Ox08. The 2nd time, value at address Ox200+2=Ox202 should add by 1, Ox02 + 1 = Ox03. The 3rd time, value at address Ox200+3 = Ox203 should add by 1, Ox0F+1=Ox10.
Problem 5: What does “loopEnd: jmp loopEnd” do?

infinite loop

Problem 6: Fill in the blanks with the appropriate addressing mode names:

<table>
<thead>
<tr>
<th>Address</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xC000</td>
<td>0x 07 E0</td>
</tr>
<tr>
<td>0xC002</td>
<td>0x 02 0F</td>
</tr>
<tr>
<td>0xC004</td>
<td>0x 14 11</td>
</tr>
<tr>
<td>0xC006</td>
<td>0x</td>
</tr>
<tr>
<td>0xC008</td>
<td>0x</td>
</tr>
<tr>
<td>0xC00A</td>
<td>0x</td>
</tr>
</tbody>
</table>

#0x1234 *Immediate*

&var *Absolute*

R10 *Register*

var(R10) *Index*

Problem 7:

What preprocessor directives do you use to define a symbolic constant? Where do you put the symbolic constant definition in your code? Where do symbolic constants go in memory?

“.set”

Anywhere

Symbolic constants do not go into memory.

Problem 8:

Describe what does the following code do? (You do not need to describe the peripheral behavior)

bis.b #BIT0 + BIT6, &P1DIR
xor.b #BIT6, &P1OUT
Problem 9: How many core registers is there in the MSP430G2553 CPU? How many of them are general purpose?

16 core registers, 12 general purpose registers, 4 special registers

Problem 10: What is the difference between using the hammer button and the bug button in code composer studio?

bug tool also downloads program into MCU.

Extra Credit: Instruction Timing

A screenshot of the disassembler view in code composer studio is as shown below. How many clock cycles do the instructions at 0xC00A (5) and 0xC01A (5) take respectively?

c00a: D0F2 0041 0022   BIS.B #0x0041,&Port_1_2_P1DIR
c010: E0F2 0040 0021   XOR.B #0x0040,&Port_1_2_P1OUT

myfun:
c016: E3D2 0021   XOR.B #1,&Port_1_2_P1OUT

c01a: 40B2 000A 0200   MOV.W #0x000a,&0x0200

loop:
c020: 8392 0200   DEC.W &0x0200
c024: 23FD   JNE (loop)
c026: 3FF7   JMP (myfun)