EE 308

Exam 2

March 30, 2005

Name: _____

You may use any of the Motorola data books and the notes from the web. Show all work. Partial credit will be given. No credit will be given if an answer appears with no supporting work.

For all the problems in this exam, assume you are using an HCS12DP256 with a 4 MHz crystal and a 24 MHz bus clock.

Also, assume that `iodp256.h` has been included, so you can refer any register in the HCS12 by name rather than by its address in any C code you write.

1. The following questions concern writing C code.

   (a) Write some C code which will set bits 2 and 3, and clear bits 0 and 7 of the byte at address 0x2400, and leave the other bits unchanged.

   ```
   *(char *) 0x2400 = ((* (char *) 0x2400) | 0x0c) & ~0x81;
   ```

   or

   ```
   #define ADDR (*(char *) 0x2400)
   Addr = (ADDR | 0x0c) & ~0x81;
   ```

   (b) Write some C code which will write the 16 bit number 0x55aa to the word at address 0x2402.

   ```
   *(int *) 0x2402 = 0x55aa;
   ```

   (c) Write some C code which will wait until bit 2 of the TFLG1 register becomes one.

   ```
   while ((TFLG1 & 0x04) == 0) ;
   ```

   or

   ```
   while (!(TFLG1 & 0x04)) ;
   ```

2. Below are the contents of the memory of an HCS12:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
10D0	10	23	3B	7C	10	04	86	80	B7	10	25	3B	FC	10	18	F3
10E0	12	50	FD	10	18	86	40	B7	10	23	3B	FC	10	12	DD	02
10F0	86	02	B7	10	23	3B	7C	10	03	86	40	B7	10	25	3B	86
FFC0	CC	05	9F	CD	99	03	84	9C	01	9B	CC	90	66	FC	93	30
FFD0	7E	E3	4B	7E	E5	38	21	54	05	83	10	34	2A	38	3C	03
FFE0	41	38	66	F2	7C	13	37	1C	25	F2	1C	38	5F	1B	42	1A
FFF0	1A	26	21	13	6A	AA	20	1F	4B	38	33	38	45	38	10	20

(a) What is the address of the first instruction the HCS12 will execute when coming out of reset?

The reset vector is at address 0xFFFE. This 16-bit location has an 0x1020. This is the address of the first instruction the HCS12 will execute when coming out of reset.

(b) What is the address of the first instruction of the Timer Channel 2 interrupt service routine?

The interrupt vector for Timer Channel 2 is at 0xFFEA. This 16-bit location has an 0x1c38. This is the address of the first instruction of the TImer Channel 2 interrupt service routine.

3. Below are the contents of the memory of an HCS12:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
10D0	10	23	3B	7C	10	04	86	80	B7	10	25	3B	FC	10	18	F3
10E0	12	50	FD	10	18	86	40	B7	10	23	3B	FC	10	12	DD	02
10F0	86	02	B7	10	23	3B	7C	10	03	86	40	B7	10	25	3B	86
FFC0	CC	05	9F	CD	99	03	84	9C	01	9B	CC	90	66	FC	93	30
FFD0	7E	E3	4B	7E	E5	38	21	54	05	83	10	34	2A	38	3C	03
FFE0	41	38	66	F2	7C	13	37	1C	25	F2	1C	38	5F	1B	42	1A
FFF0	1A	26	21	13	6A	AA	20	1F	4B	38	33	38	45	38	10	20

The HCS12 registers have the following values when an unmasked Real Time Interrupt occurs:

Reg	-	-
	S X H I N Z V C	
CCR	1 1 1 0 1 0 0 1	
A:B	A3	92
X	A51C	
Y	2020	
SP	3BE5	
PC	1024	

(a) Explain in detail what happens when the HCS12 responds to the interrupt.

The HCS12 completes the current instruction. It then stacks the information it needs to save: the return address, the Y, X, B, A and Condition Code registers. It sets the I bit of the CCR (and the X bit, if it is the XIRQ interrupt), then loads the Program Counter with the address from the interrupt's vector.

(b) Show what will be in the HCS12 registers when it starts executing the first instruction of the interrupt service routine.

Reg	-	-
	S X H I N Z V C	
CCR	1 1 1 1 1 0 0 1	
A:B	A3	92
X	A51C	
Y	2020	
SP	3BDC	
PC	1A26	

The interrupt was a Real Time Interrupt, whose vector is at 0xFFF0, which contains an 0x1A26. Thus, the PC will be loaded with an 0x1A26. The I bit of the CCR will be set. The stack pointer will be decremented by 9, from 0x3BE5 to 0x3BDC.

(c) Also, show what has happened to the stack – fill in values for memory locations which have changed below.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3BD0													E5	92	A3	A5
3BE0	1C	20	20	10	24											
3BF0																

- The SP is decremented by 2 to 0x3BE3; the return address (0x1024, the PC when the interrupt is received) is pushed.
- The SP is decremented by 2 to 0x3BE1; the Y register is pushed.
- The SP is decremented by 2 to 0x3BDF; the X register is pushed.
- The SP is decremented by 2 to 0x3BDD; the registers B and A are pushed.
- The SP is decremented by 1 to 0x3BDC; the Condition Code Register value (before the interrupt; i.e., with the I bit equal to 0) is pushed.

(d) List at least five things needed in your program when you use interrupts.

- Load the Stack Pointer.
- Have an interrupt service routine (ISR).
- The ISR must clear the source of the interrupt before exiting.
- The ISR must exit with an RTI (Return beginfrom Interrupt) instruction.
- Do whatever setup is necessary to initialize the interrupt hardware (for example, start the timer and set the prescaler if using the Timer Overflow Interrupt).
- Make sure the Interrupt Vector is loaded with the address of the first instruction of the ISR.
- Enable the specific interrupt (for example, for the Timer Overflow Interrupt, set the TOI bit of the TSCR2 register).
- Enable interrupts in general by clearing the I bit of the CCR (with the CLI instruction).

(e) The assembly language Real Time Interrupt interrupt service routine is the following:

```
rti_isr:
            inc PORTA
            rti
```

How much time (in bus cycles and in seconds) does it take for the HCS12 to respond to the Real Time Interrupt – i.e., from the time the HCS12 receives a Real Time Interrupt, how long does it take for the HCS12 to get into, execute, and exit from the interrupt service routine? Explain.

- It takes 9 cycles to start executing the ISR. (See Figure 6.1 on Page 132 of the HCS12 V1.5 Core User Guide to see what the HCS12 does on each cycle, or the description of the SWI intruction).
- The `inc PORTA` instruction takes 4 cycles (the instruction uses the extended addressing mode)
- The `rti` instruction takes 8 cycles (or 11, if another interrupt is pending).

The total number of cycles is $9 + 4 + 8 = 21$ (or $9 + 4 + 11 = 24$ if another interrupt is pending). Each cycle takes $1/24 \times 10^{-6}$ seconds (with a 24 MHz bus clock), so the total time is $21/24 \times 10^{-6} = 0.875$ μs, (1 μs if another interrupt is pending).

(f) What is wrong the the above interrupt service routine?

The routine does not clear the source of the interrupt. It should write a 1 to the RTIF (real time interrupt flag) of the CTFLG register. The ISR should be:

```
rti_isr:
            inc PORTA
            RTIF = 0x80;
            rti
```

4. The following questions pertain the the HCS12 timer subsystem.

   (a) How do you enable the HCS12 timer subsystem? Write some C code to do this.
   Write a 1 to the TEN bit of the TSCR1 register:

   ```
   TSCR1 = TSCR1 | 0x80;
   ```

   (b) What is the basic frequency of the timer subsystem clock – i.e., the frequency before changing the prescaler?
   For our HCS12, the basic frequency is 24 MHz.

   (c) How do you change the frequency of the timer subsystem clock? Write some C code to set the frequency to 3 MHz.
   You need to divide the 24 MHz basic frequency by 8. You do this by setting the prescaler (PR2:0 of TSCR2) to 011:

   ```
   TSCR2 = (TSCR2 | 0x03) & ~0x04;
   ```

   (d) Write some C code to clear `C4F`, the flag for timer channel 4. Be sure your code does not clear any other timer flag which may be set.
   Write a 1 to the `C4F` bit of the TFLG1 register, and 0's to all other bits:

   ```
   TFLG1 = 0x10;
   ```

   (e) Write some C to set up timer channel 4 to function as an input capture. Set it up to capture a falling edge. Be sure that you do not change the functionality of any other timer channel.
   Write a 0 to Bit 4 of TIOS register to make Channel 4 an input capture. Set `EDG4B:EDG4A` of TCTL3 to 01:

   ```
   TIOS = TIOS & ~0x10;
   TCTL3 = (TCTL3 | 0x01) & ~0x02;
   ```

   (f) Write some C to set up timer channel 5 to function as an output compare. Set it up to set Channel 5 output high on a successful compare. Be sure that you do not change the functionality of any other timer channel.
   Write a 1 to Bit 5 of TIOS register to make Channel 5 an output compare. Set `OM5:OL5` of TCTL1 to 11:

   ```
   TIOS = TIOS | 0x20;
   TCTL1 = TCTL1 | 0x0c;
   ```

5. The HCS12 is being used to control the intensity of a light. The light needs a PWM frequency of 2.5 kHz. Write some code which will enable PWM Channel 2 with a 2.5 kHz frequency and a duty cycle of 25%.

   Want 2.5 kHz frequency. The basic frequency is 24 MHz. 24 MHz/2.5 kHz = 9,600, so need to divide the 24 MHz clock by 9,600 to get 2.5 kHz.

   Channel 2 uses `PCKB` and `PWMSCLB`.

   - Using clock mode 0, want 9,600 = $2^{\texttt{PCKB}} \times$ `PWMPER2`. One way to do this is to set `PCKB` to 6 (so $2^{\texttt{PCKB}} = 64$) and `PWPMER2` to 150.
   - Using clock mode 1, want 9,600 = $2^{\texttt{PCKB + 1}} \times$ `PWPMER2` $\times$ `PWMSCLB`. One way to do this is to set `PCKB` to 0 (so $2^{\texttt{PCKB + 1}} = 2$), `PWMPER2` to 200, and `PWMSCLB` to 24.

   I will use clock mode 1.

   Do the following:

   - Choose 8-bit mode (PWMCTL = 0x00)
   - Choose high polarity (PWMPOL = 0xFF)
   - Choose left-aligned (PWMCAE = 0x00)
   - Set PCLK2 = 1 in PWMCLK
   - Set PCKB = 0 in PWMPRCLK
   - Set PWMSCLB = 24
   - Set PWMPER2 = 200
   - Enable PWM Channel 2 (set bit 2 of PWME)
   - For 50% duty cycle, set PWMDTY2 = 50% $\times$ PWMPER2 = 50% $\times$ 200 = 100

```
PWMCTL = 0x00;                  /* 8-bit mode */
PWMPOL = 0xFF;                  /* high polarity */
PWMCAE = 0x00;                  /* left-aligned */
PWMCLK = PWMCLK | 0x04;         /* Clock mode 1 for Ch 2 */
PWMPRCLK = PWMPRCLK & ~0x70;    /* PCKB = 0 */
PWMSCLB = 24;
PWMPER2 = 200;
PWME = PWME | 0x04;             /* Enable PWM Ch 2 */
PWMDTY2 = 100;
```