Christopher Badolato
EEL 4742 C 0002
9/25/2019

Homework 1

Question 1

Part a) A memory is byte addressable and has a 17-bit address. All the addresses are valid. What is the total size of the memory?

$2^{17} = 2^{10} \cdot 2^{7} = 128\text{KB}$

Part b) A memory is byte addressable and has a total size of 18,432 Bytes (18KB). What is the smallest address size that can be used for this memory.

10 bit => 1KB
11 bit => 2KB
12 bit => 4KB
13 bit => 8KB
14 bit => 16KB
15 bit => 32KB ←

15 bit will be the smallest address size that can be used.

Must be larger than total size

Homework 1

Question 2

A) A computer maintains memory alignment. At what address can we store a byte variable? What about a 16-bit varinble?

For	Aligned @
8-bit data	multiples of 1
16-bit data	multiples of 2
32-bit data	multiples of 4

B) A computer maintains memory alignment. Show how the variables below are stored in memory if they have to be stored in the order they are declared (x, f, y, g, z) starting at address 500.

```
unsigned char x;    // 8-bit
short int f;        // 16-bit
unsgned char y;
short int g;
unsigned char z;
```

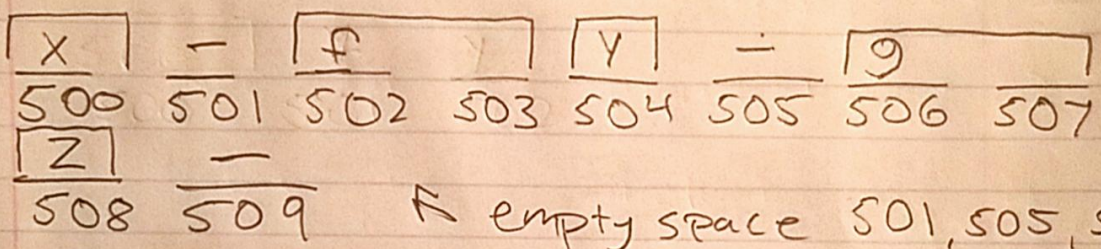
C) Repeat above knowing that the memory should be aligned but the variables can be stored in any order

Work for B and C on Next page

Question 2

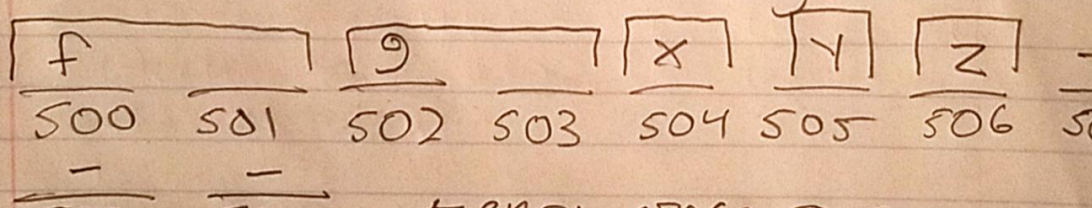
B)
```
unsigned char x;  // 8-bit
short int f;      // 16-bit
unsigned char y;
short int g;
unsigned char z;
```

Stored starting at 500



← empty space 501, 505, 509

C) Variables stored in any order



← empty space @ 507, 508, 509

Homework 1

Question 3

A) Explain the Big Endian and the Little Endian configurations

Big Endian: Most significant Byte at the Lowest address

Little Endian: Most significant at the highest address.

0x12EF

B) Big

12	EF
500	501

Little

EF	12
500	501

show how the data (0x12EF) is stored at address 500 in the two configurations

C) which Configuration is used in the MSP430?

MSP 430 use Little Endian

Homework 1

Question 4

A) A microcontroller's memory map allocates the FLASH code space to the address range [0x0400] to [0x0BFF]. What code size in Bytes will be supported by this microcontroller?

0000 0100 0000 0000 2^{10} = 1024

to 0000 1011 1111 1111 = 3071

$2^{11}+2^{9}+2^{8}$

3071

-1024

2047 ⇐ 2048 total memory Locations

Each Location can store 2 Bytes of Data

Code size = 2048 x 2 Bytes

= 4096 Bytes or 4kB

B) The Vector table contains memory addresses (A Vector is a memory address). In certain MSP430 device, the Vector table is in the range [0xFFC0 to 0xFFFF]. The memory address is 16-bit. How many Vectors does this table support?

1111 1111 1100 0000 65472 65535

1111 1111 1111 1111 65535 -65472

63

63+1 = 64 Vectors

HW1

Question 5

A) Two clock technologies used in microcontrollers are the crystal oscillator and the RC oscillator. For each of them, comment on:

	XT (crystal)	RC oscillator
Startup speed	up to 10^5 dead cycles to stabilize	10 cycles or Less to stabilize
Accuracy	Parts per million XT is more Accurate	low accuracy are senstative to EMI, Vibrations and noise
Stability with temp and Voltage Variations	more stable they have lower Voltage Variations	Poor temp stability and Voltage Variation
price	They have low cost crystals but the cost to performance of a more expensive XT will have better performance	Low cost but have performance isshies and are susceptible to Noise

HW 1

Question 6

A) Why does embedded programming use extensions to C?

In C, the embedded programs will be executed by the embedded processors, and provide portability "@ operator"

B) Downside of extensions regarding code probability?

- extensions are compiled extensions

C) what is an intrinsic function?

- tell the compiler which function to use. "hints for the compiler."

D) In MSP 430, interrupts are enabled/disabled by writing to the bit called GIE in the status register (SR; R2) why do we use.

- enable_interrupts();
- disable_interrupts();

- An interrupt Lanches an Interrupt service routine.

Benefits:

- CPU doesn't have to poll the event
- Enables Fast response to Interrupts

E.g. Overflow.

Hw1

Question 7

A) How many bits is an int (integer) in the C language?

16-bits or 32 bits

B) why does embedded programming use data types like (uint_8t, int_8t, uint_16t, int_16t)

- So that we can select the size or type of integer we need to store a certain amount of Bits

C) unsigned Int counter from 0 to 45,000 to create a small delay. If we double the delay, can they simply change the value to 90,000?

No, the counter speed depends upon the workload of the CPU and resource Availability.

D) If the code contains a delay loop and we notice no delay is being created at runtime. What should we suspect during debugging?

- Check if loop is being satisfied.
- Check initial value of loop variable.
- Check for break or exit statement.

Hw1
Questions 8 and 9

I typed these two Questions out since there was a lot of writing.

ON NEXT PAGE

QUESTION 8. (10 points)

For the questions below, write the code using the masks that are pre-defined in the header file. (E.g: BIT0:0000 0001, BIT1:0000 0010, …, BIT7:1000 0000).
Perform the operations below on the 8-bit variable (uint_8t data).

Part a) Write code that performs the three operations below. Perform each operation independently of the other

Set bit 6

```
data = BIT6;
```

• Clear bit 6.

```
data &= ~BIT6;
```

• Invert bit 6.

```
data ^= BIT6;
```

Part b) Write code that performs the three operations below. Perform each operation independently of the others.

• Set bits 4 and 5.

```
data = (BIT4 | BIT5);
```

• Clear bits 4 and 5.

```
data &= ~(BIT4 | BIT5);
```

• Invert bits 4 and 5.

```
data ^= (BIT4 | BIT5);
```

• Set bit 4 and clear bit 5.

```
data |= BIT4;
data &= ~BIT5;
```

Part c) Write an if-condition line for each of the cases below. Perform each operation independently of the others.

• Check if bit 2 is 1.

```
If ((data & BIT2) == BIT2){

}
```

• Check if bit 2 is 0.

```
If ((data & BIT2) == 0){
```

```
}
```

• Check if bits 3,4 are 1,1.

```
If ((data & (BIT3|BIT4)) == (BIT3|BIT4)){

}
```

• Check if bit 3 is 0 and bit 4 is 1.

```
If ((data & (BIT3|BIT4)) == BIT4){

}
```

• Check if bits 3, 4 are 0,0.

```
If ((data & (BIT3|BIT4)) == 0){

}
```

QUESTION 9. (10 points)

A module on the microcontroller is configured using a control register called CTL that has the format shown below.

SLP	CLK	CAP	IE
2-	3-	2-	1-

- SLP: selects sleep mode; value between 0 and 3
- CLK: selects clock speed; value between 0 and 7
- CAP: selects built-in capacitor value; choice between 0 and 3
- IE: interrupt enable bit (1: enable/ 0: disable)

To support programming the device, the environment has declared the symbolic constants:

```
SLP_3: 1100 0000
SLP_2: 1000 0000
SLP_1: 0100 0000
SLP_0: 0000 0000

CLK_7: 0011 1000
CLK_6: 0011 0000
…
CLK_0: 0000 0000
```

```
CAP_3: 0000 0110

CAP_0: 0000 0000

IE: 0000 0001
```

Perform all the operations below using the masks defined above.

Part a) Write a line of code that configures the module as the following:

```
(Sleep mode 2)(Clock speed 6)(Capacitor value 1)(Interrupts enabled)
```

```
CTL &= (SLP_2|CLK_6|CAP_1|IE);
```

Part b) For the operation above, show the masks used and the final value of CTL in binary.

SLP_2 = 1000 0000
CLK_6 = 0011 0000
CAP_1 = 0000 0010
IE = 0000 0001

Final Value
CTL = 1011 0011

Part c) Write a piece of code that changes SLP to 1. The current value of SLP is unknown.

We first AND to clear the bits previous in the SLP, then we can OR the bits back onto the variable

```
CTL &= ~SLP_3;
CTL |= SLP_1;
```

Part d) Write a piece of code that changes CLK to 5. The current value of CLK is unknown.

We first AND to clear the bits previous in the CLK, then we can OR the bits back onto the variable

```
CTL &= ~CLK_7;
CTL |= CLK_5;
```

Part e) Write an if-condition line that checks if SLP=1.

```
If ((CTL & SLP_3) == SLP_1)){

}
```

Part f) Write an if-condition line that checks if CLK=5.

```
If ((CTL & CLK_7) == CLK_5)){

}
```

Part g) Write an if-condition that checks if the current value of CLK is either of (0, 2, 4, 6).

```
If ((CTL & CLK_7) == CLK_0 | (CTL & CLK_7) == CLK_2 | (CTL & CLK_7) == CLK_4 |
(CTL & CLK_7) == CLK_6 | ){

}
```