CG1112 Engineering Principle and Practice II
Semester 2 2018/2019

Week of 4 March 2019
Tutorial 4 Suggested Solutions
Revision for Week 1 to 6

This tutorial serves both as a revision and also a sampler for the "midterm question style".

1. [Raspberry Pi] Which of the following statement(s) regarding the difference between Raspberry Pi and a typical PC is/are TRUE?
    i. Pi uses ARM processor while PC typically uses Intel processor.
    ii. Pi does not use RAM for runtime storage, unlike typical PC.
    iii. Pi does not have persistent storage, unlike typical PC.

    a. (i) only.
    b. (i) and (ii) only.
    c. (ii) and (iii) only.
    d. (i), (ii) and (iii).
    e. None of the above.

ANS:

Pi is a miniature PC, which the same main components: processor, RAM and secondary (persistent) storage. So, only statement (i) is correct.

Question 2-4 uses the following context:

Given an array A with N unsorted numbers and a target sum S, we want to find out whether there are two numbers in A that adds up to S. i.e. A[i] + A[j] == S. This is commonly known as the **pair-sum problem**.

2. If we solve the pair-sum problem by **exhaustively trying out all pairs of number**:

```
0; i < N-1; i++){
(j = i+1; j < N; j++){
f (A[i] + A[j] == S)  return (i, j);   //pseudo-code
```

The **worst-case** time complexity for this approach is:

a. O(1)
b. O($\log_2(N)$)
c. O(N)
d. O(N $\log_2(N)$)
e. O(N^2)

ANS:

The if- statements get executed N-1, N-2, N-3….. 1 times ➔ O(N^2)

3. If we solve the pair-sum problem by:

```
A:  bubble sort the array A,
B:  for each A[i], binary search for (S-A[i])
```

What is the most accurate time complexity of this approach?

a. Phase A = O(N^2); Phase B = O($\log_2(N)$)
b. Phase A = O(N $\log_2(N)$); Phase B = O($\log_2(N)$)
c. Phase A = O(N^2); Phase B = O(N $\log_2(N)$)
d. Phase A = O(N $\log_2(N)$); Phase B = O(N $\log_2(N)$)
e. None of the above

ANS:

Bubble sort is O(N^2)
Each of the binary search is O($\log_2(N)$), but we potentially need to search for ½ of the N elements ➔ O(N $\log_2(N)$).

So, (C) is the answer.

4. If we modify the solution in (3) by

```
A:  bubble sort the array A,
B:  set left = 0, right = N-1
     while (left < right) {
          if (A[left] + A[right] == S)  return (left, right);
```

```
if (A[left] + A[right] > S)    right--;
else if (A[left] + A[right] < S)    left++;
```

What is the most accurate time complexity of the **phase B**?

a. O(1)
b. O($\log_2(N)$)
c. O(N)
d. O(N $\log_2(N)$)
e. O(N^2)

ANS:

Every iteration the left OR right will move one step. The code terminates when left and right meets ➔ Worst case the whole array is tested ➔ O(N)

5. [Git] Which of the following statements regarding the **staging** process of the Git is TRUE?

    a. If a file is already being tracked by Git, there is no need to stage the file to commit new change.
    b. To add a new file for Git to track, there is no need to stage the file to commit new change.
    c. We always need to stage **all modified files** before committing.
    d. Forgetting to stage a file will cause **merge conflict.**
    e. None of the above

ANS:

(a) to (c) are the opposite facts. (d) is simply made up ☺. Answer = (e)

6. Which of the following shows the correct sequence for interrupt processing?

(Note: PC = Program Counter, which keeps track of which instruction the CPU will execute next)

a. Detect interrupt, get address of ISR, execute ISR, pop PC from stack, push PC to stack.
b. Get address of ISR, detect interrupt, push PC to stack, get address of ISR, execute ISR, pop PC from stack.
c. Push PC to stack, detect interrupt, get address of ISR, execute ISR, pop PC from stack.
d. Detect interrupt, get address of ISR, push PC to stack, execute ISR, pop PC from stack.
e. Detect interrupt, pop PC from stack, get address of ISR, execute ISR, push PC to stack.

Option d is correct: detect the interrupt, find the ISR for that interrupt, push PC to save it, execute the ISR, pop PC to restore execution to the previous code.

Option a. means that we lose the contents of PC, which prevents us from resuming execution of the interrupted code. Option b. doesn't make sense; how do we decide which ISR if we don't know which, if any interrupt was triggered? Option c is possible but if there's no interrupt then you need to pop PC. Option e will result in randomness being written to the PC when you pop before pushing it.

7. We have two buttons B1 and B2 connected to INT0 and INT1 respectively. EICRA is set so that INT0 and INT1 are triggered by the rising edge of the signal. EIMSK is set to enable both INT0 and INT1.

The following code is executed on an Atmega328P:

```
static volatile int turn = 0;
ISR(INT0_vect)
{
        turn=0;
}

ISR(INT1_vect)
        {
                turn =1;
        }
```

```
int main()
{
        while(1)
        {
                // flashRed and flashGreen flash the red and green LEDs
                // at a rate of 1 Hz, respectively.
                if(turn == 0)
                        flashRed();
                else
                        flashGreen();
        }
}
```

We press both A and B at exactly the same time, then release B, before releasing A. Which statement best describes what will happen?

a. The green LED will flash as long as B is held down, and the red LED will flash once B is released.
b. The red LED will flash while B is held down, and the green LED will flash once B is released.
c. The red LED will flash momentarily, then the green LED will flash.
d. The red LED will flash until button A is released, then the green LED will flash.
e. The green LED will flash momentarily, then the red LED will flash.

The answer is c. This is because INT0 and INT1 will be triggered at the same time, but INT0 has a higher priority than INT1, so flashRed is called. However once INT0_vect ISR is completed, control is handed to INT1_vect ISR, which will cause the green LED to flash.

8. Examine the following code snippet.

```
int a = 10, b = -10, c;
c = a && b;
a = (5 || !3) && 6;
```

What is the value of c and after the code is executed?

c = 1 and a = 1

The code shows logical operations. Each variable represents either TRUE '1' or FALSE '0'. A value of zero in the variable will classify its state as FALSE. Any non-zero value in the variable will classify it as TRUE (the actual value doesn't matter).

9. Given the following assignment <**char x = 0xA1;**>, write a code snippet that will assign the lower 4 bits of 'x' to variable 'lower_x' and the upper 4 bits to variable 'upper_x'.

After the code is executed the values in the variables will be as such:

```
lower_x = 0x1;
upper_x = 0xA;

lower_x = (x & 0x0F);
upper_x = (x & 0xF0) >> 4;
```

10. Two switches are connected to PORT B pins 0 and 5. The switch connected to Pin 0, is operating in Active-High mode (a logic '1' is read when the switch is pressed, '0' otherwise). The switch connected to Pin 5 is operating in Active-Low mode (a logic '0' is read when the switch is pressed, '1' otherwise). An Active-Low LED is connected to PORTD Pin 7 (a logic '0' turns it ON and a logic '1' turns it OFF).

Write a code-snippet that will:

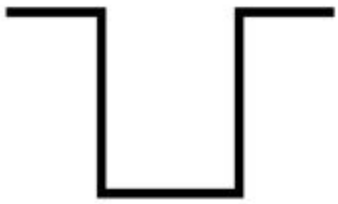
- Turn ON the LED if **either one** Switch is pressed.
- Turn OFF the LED if **both** switches are pressed or not pressed.

```
#define MASK_BIT_7 0b10000000
#define MASK_BIT_5 0b00100000
#define MASK_BIT_0 0b00000001

#define MASK BIT
DDRB &= ~(( MASK_BIT_5) | (MASK_BIT_0)) ;
DDRD |= (MASK_BIT_7);

if ((PINB & MASK_BIT_0) && !(PINB & MASK_BIT_5))    // Turn OFF LED
{
        PORTD |= (MASK_BIT_7);
}
else if ((PINB & MASK_BIT_0) || !(PINB & MASK_BIT_5)) // Turn ON LED
{
        PORTD &= ~(MASK_BIT_7);
}
else // Turn OFF LED
{
        PORTD |= (MASK_BIT_7);
}
```

11. Timer 0 is to be operated in Counter Mode to keep track of the number of times an external event (e.g. a switch press) occurs. The event generates a 10us pulse (as shown below) when triggered.



a. What should be the Clock Select Bits in order for the Timer Block to capture the event the moment it is triggered?

Since the event is characterized by a negative transition pulse, we must capture the falling edge of the signal. The Clock Select Bits must be configured as '110'. These are bits [3:0] of TCCR0B.

1	1	0	External clock source on T0 pin. Clock on falling edge.

b. Which pin on the AT238p must be used to capture the signal?

The TO pin is tied to Port D Pin 4, which is mapped to Arduino Pin 4.

c. You wish to trigger an interrupt whenever 5 such pulses are detected by the microcontroller. Explain how you can configure the TCNT0 and OCR0A registers to achieve this.

We need to set the values in these registers to be 4 counts apart.
TCNT0 = 0;
OCR0A = 4;

We also need to enable the OCIEA bit in the TIMSK0. When the counter counts from 0 to 4, it will show a match, but the interrupt is only triggered in the following clock cycle. So, a total of 5 events will be captured.

12. You decide to operate Timer0 in Fast PWM mode. The WGM0[2:0] bits are set to 0x3. What is the frequency of the PWM signal with the prescaler set to 1.

When the WGM0[2:0] are set to 0x3, the TOP is defined as 0xFF.
Freq = 16M / 256 = 62.5 kHz

13. While operating in Fast PWM mode, you decide NOT to use interrupts.
    a. Which Register/Bit must you check in order to know that there is a match between TCNT0 and OC0RA?

    The OCFA (bit 1) in the TIFR0 register will be set when there is a match between TCNT0 and OCR0A.

    b. Write the code that will clear the OCFA bit.

    To clear the flag, we must write a '1' to it.
    TIFR0 |= 0b00000010;