NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

SEMESTER II AY2017/2018

MIDTERM ASSESSMENT FOR

CG1112: ENGINEERING PRINCIPLES AND PRACTICE II

10th March 2018 Time Allowed: 1 Hour

INSTRUCTIONS TO CANDIDATES:

1. Use **2B Pencil** to shade the **OCR form**. Ensure your student number is shaded properly.

2. This assessment paper consists of **TWENTY (20)** questions.

3. This assessment paper comprises **THIRTEEN (13) printed pages** including this front page.

4. Each MCQ carries 1 mark. No penalty for wrong answer. Total marks for the paper is **20**.

5. This is a **close book assessment** with **one A4 reference sheet** allowed.

6. Submit only the **OCR** form at the end of the assessment.

CG1112 Midterm AY1718S2

1. Give the tightest Big-O complexity for the following code fragment:

```
i = 1;
j = 0;
while (j < N ){
    if (i % N == 0){
       j++;
       <5 operations>
    }
    i++;
}
```

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

Commented [ys1]: Answer. This is a flatten loop. "j" is increased for every N increment of "i". So, "i" is increased (N+1) x N times in total.

2. Give the tightest Big-O complexity for the following function `f()`:

```
void f( int N )
//N is a non-negative integer
{
     int i;
     while (N > 0) {
         for (i = 0; i < N % 10; i++){
             <5 operations>
         }
         N /= 10;
     }
}
```

For convenience, we define **D** as the number of digits in **N.**

   a. O(N)
   b. O(D)
   c. O(N x D)
   d. O(N^2)
   e. O(D^2)

Commented [ys2]: while-loop runs D times.
Each for-loop runs at most 9 times (as it is based on the digit value [0 to 9].
So, O(9D) ➔ O(D)

3. If we are told that algorithm A is of complexity O(N) and algorithm B is of complexity O(N $\log_2$(N)), which of the following statements is/are TRUE?
    i. A runs faster than B for all N.
    ii. If an algorithm C uses both A and B, then C has the same complexity as B.
    iii. When input size N is larger than a specific value N_x, algorithm A is always slower than algorithm B.

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

Commented [ys3]: The only true statement.
C = O(N) + O(N lg N) = O(N lg N)

Commented [ys4]: Answer

4. Which of the following statements regarding Pi is/are TRUE?
    i. Pi is a mini-computer while ATmega328P is a micro-controller.
    ii. Raspbian is a boot loader program for Pi.
    iii. Pi supports only C/C++ programming.

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

Commented [ys5]: The only true statement. Essentially give away (hopefully) ☺

Commented [ys6]: Answer.

5. Which of the following statements regarding Git is/are TRUE?
    i. There can be multiple Git repositories on the same hard disk.
    ii. Even with a single user, it is possible to encounter merge conflict when pushing to a remote repository.
    iii. If a folder is part of a Git repository, then all files and sub folders under it are part of the same repository too.

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

Commented [SYJ7]: True. Git operates at folder level.

Commented [SYJ8]: True. Example: single user using multiple devices and didn't pull his/her committed changes from other devices properly.

Commented [SYJ9]: False. Example: non-staged file or gitignore files.

Commented [SYJ10]: Answer.

6. Suppose our good friends Uno and Duo are working on a project with two files `Midterm.txt` and `Easy.txt`. The project files are stored on a remote Git repository. You can assume that both Uno and Duo started out with a clean local clone on their respective laptop. Identify all sequence of working that results in a **merge conflict that requires manual intervention**.

i.	Uno edits [***Easy.txt, line 12***]; Duo pulls then edits [***Easy.txt, line 12***]; Uno commits and pushes; Duo commits and pushes.
ii.	Duo edits [***Easy.txt, line 12***]; Uno edits [***Midterm.txt, line 12***]; Uno commits and pushes; Duo commits and pushes.
iii.	Duo edits [***Midterm.txt, line 12***]; Duo commits and pushes; Uno pulls then edits [***Midterm.txt, line 12***]; Uno commits and pushes;

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

Commented [ys11]: The only true statement. Note that the pull by Duo is useless as Uno has not yet committed.

Commented [ys12]: Will have a merge warning, but can be automatically resolved as the changes are not related.

Commented [ys13]: The pull by Uno prevented the conflict.

Commented [ys14]: Answer.

7. You decide to operate the AT328p using the Internal 128KHz RC Oscillator. When operating Timer 0 in Phase-Correct PWM mode, what is the largest period of the PWM signal using only the internal pre-scaler.

   The list of prescalers available within the microcontroller are 1, 8, 64, 256 and 1024.

a. 4.08 s
b. 0.245 s
c. 7.8 us
d. 8 ms
e. 10 ms

Commented [SYJ15]: Answer

8. An LED is connected to PORTB Bit 0 in Active-High configuration, with the DDRB register already configured accordingly.

   Examine the following code snippet to control the LED. Note that the initial value of variable 'x' is not yet decided.

```
char x = <to be decided>;
x = (((x & 0xF0) >> 4) | ((x & 0x0F) << 4));
if(x & 0b1)
    PORTB |= 0x01;
else
    PORTB &= ~(0x01);
```

   Which option has all possible values of 'x' that will always turn ON the LED.

   a. 0xC1, 0x1A, 0x21
   b. 0x11, 0xD1, 0x9A
   c. 0xFF, 0xAA, 0x88
   d. 0xA1, 0x11, 0x1C
   e. None of the above.

Commented [SYJ16]: Answer

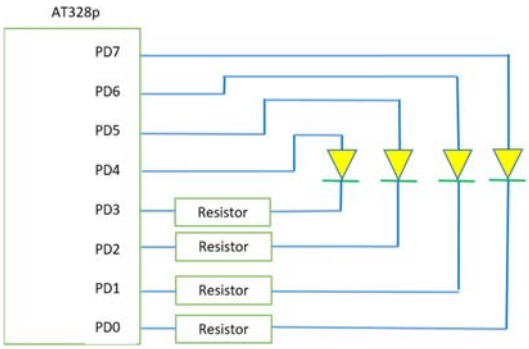
9. While operating Timer 0 in Phase-Correct PWM mode, what is the smallest equivalent analog voltage that can be achieved with the module?

   NOTE: Assume the AT328P is operating with a 5V supply.
   The list of prescalers available within the microcontroller are 1, 8, 64, 256 and 1024.

   a. 0.392 mV
   b. 19.6 mV
   c. 1.96 mV
   d. 3.92 mV
   e. 8.96 mV

Commented [SYJ17]: Answer

10. The figure below shows the connections to PORT D of the AT328P.



Which combination of DDRD and PORTD values will make all LED's be turned ON together at the same time?

NOTE: For the DDRD register, a '1' in a bit position makes it an Output and a '0' in a bit position makes it an Input.

a. DDRD = 0x0F; PORTD = 0xF0;
b. DDRD = 0x0F; PORTD = 0x0F;
c. DDRD = 0xFF; PORTD = 0x0F;
d. DDRD = 0xFF; PORTD = 0xF0;
e. DDRD = 0xF0; PORTD = 0xFF;

Commented [SYJ18]: Answer

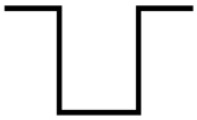
11. In your application, you are going to use all THREE Timers. If all the timers triggered their Interrupts at the same time while you were in the main() code, how many context switches would occur before the code returns to main().

NOTE: Assume that Nested Interrupts are Disabled.

a. 3
b. 4
c. 5
d. 6
e. 7

Commented [SYJ19]: Answer

12. Timer 0 is to be used in Counter Mode to keep track of the number of pulses sent by the wheel encoder. Every 10 degrees turn of the wheel generates a 10us pulse (as shown below).



CA02	CA01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	clkI/O/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

What should be the setting of the CS[2:0] bits to capture this event the moment it occurs?

a. CS[2:0] = 0b000
b. CS[2:0] = 0b001
c. CS[2:0] = 0b110
d. CS[2:0] = 0b111
e. CS[2:0] = 0b011

Commented [SYJ20]: Answer

13. With reference to the question above, what are the other possible Interrupts that can be used to achieve the same objective of capturing the number of occurrences of the encoder pulse?

a. INT0
b. INT1
c. Timer2
d. All of the above
e. None of (INT0, INT1, Timer2) is suitable.

Commented [SYJ21]: Answer

Question 14-16 use the following context:

We consider a system where we need to transfer 1 MiB (1,048,576) bytes of data across a channel that has a transfer rate of 10^6 bytes per second. Our system clock is 32 MHz. The system has a DMA channel that requires 250 cycles to set up, and the interrupt service routine (ISR) that is triggered at the end of the DMA transfer takes approximately 120 cycles to execute.

14. Suppose we use polling to transfer one character at a time. Approximately how many clock cycles are spent on polling between successive bytes?

    a. Approximately 29 cycles.
    b. Approximately 30 cycles.
    c. Approximately 31 cycles.
    d. Approximately 32 cycles.
    e. Approximately 33 cycles.

Commented [TKYC22]: d. With a transfer rate of 10^6 bytes per second, the time between transfers is 10^-6 seconds. With a 32MHz clock this works out to 32 cycles.

15. Assuming that our CPU can execute one instruction per clock cycle, and that the ISR for transferring one byte at a time takes approximately 0.5 microseconds to execute, how many clock cycles does it take to transfer the entire 1 MiB block of data?

    a. Approximately 4,194,304 cycles.
    b. Approximately 8,388,608 cycles.
    c. Approximately 16,777,216 cycles.
    d. Approximately 33,554,432 cycles.
    e. There is not enough information to decide.

Commented [TKYC23]: c. 0.5 microseconds is 16 cycles. Multiply by 1048576 gives us 16,777,216 cycles

16. Which ONE of the following statements is true?

    a. It is faster to transfer 1MiB of data using interrupts than using polling.
    b. It is faster to transfer 1MiB of data using DMA than using interrupts.
    c. It takes the same amount of time to transfer 1MiB of data whether you use interrupts, polling or DMA.
    d. It requires less hardware to use DMA than either interrupts or polling.
    e. It requires less hardware to use interrupts than polling.

Commented [TKYC24]: c. It takes 1048576/1000000 seconds in total to transfer regardless of transfer type.

17. We have two push buttons, b1 and b2, connected to pins 2 and 3 (INT0 and INT1) respectively of an Arduno UNO. Both pins are pulled down to GND via 10K resistors, and pressing a push button causes the corresponding pin to go HIGH, and the pin goes back LOW when the button is released. Details of EICRA and EIMSK are given in the appendix.

    The code below is executed on the Arduino (some parts are written in pseudocode to improve readability):

```
//… Other irrelevant code …
void flashRed()
{
      while(1)
      {
            … Turn on RED LED …
            _delay_ms(500);
            … Turn off RED LED …
            _delay_ms(500);
      }
}

void flashGreen()
{
      while(1)
      {
            … Turn on green LED …
            _delay_ms(500);
            … Turn off green LED …
            _delay_ms(500);
      }
}
void setup()
{

      EICRA = 0b00000111;
      EIMSK = 0b00000011;
 }

ISR(INT0_vect)
{
      flashRed();
}

ISR(INT1_vect)
{
      flashGreen();
}

int main()
{
      setup();
      while(1);
}
```

We press both buttons simultaneously and after a second, release b1, followed shortly by releasing b2.

a. The red LED will flash permanently.
b. The green LED will flash permanently.
c. The red LED will flash while b1 is held down, and the green LED will flash once b1 is released.
d. The green LED will flash while b1 is held down, and the red LED will flash once b1 is released.
e. Neither LED will flash

Commented [TKYC25]: a. INT0 has a higher priority than INT1, and flashRed contains an infinite loop and is inside the ISR, meaning that the INT1 interrupt is never processed because it is lower priority and because by default interrupts are disabled while another interrupt is being handled.

Question 18-20 use the following context:

We have the following code running on an Atmega328P running at **12 MHz** (not Arduino UNO). Please refer to the Appendix for the configuration registers for Timer 2. Timer 2's configuration details are a little different from Timer 0.

```
void setup()
{
      DDRD |=0b01100000;
      cli();
      TCCR2A = 0b00000010;
      TIMSK2 = 0b00000110;
      TCCR2B = 0b00000011;
      TCNT2=0;
      OCR2A=64;
      OCR2B=128;
      sei();
}

ISR(TIMER2_COMPA_vect)
{
      PORTD |= 0b01000000;
}

ISR(TIMER2_COMPB_vect)
{
      PORTD &= 0b10111111;
}

int main()
{
      setup();
      while(1);
}
```

18. What prescalar are we using in this code?

    a. 1
    b. 32
    c. 64
    d. 128
    e. 256

Commented [TKYC26]: b. 32. This is tricky because the prescalars for TIMER2 are different than for TIMER0 and TIMER1. If the students blindly follow the notes they will get it wrong. NOTE: WE WILL PROVIDE DATA SHEET EXTRACT FOR TIMER 2!

19. Supposing the prescalar chosen is 128 (which may, or may not be the answer to the previous question. ;)), what is the rate at which TCNT2 is incremented?

    a. Once every 3 microseconds.
    b. Once every 4 microseconds.
    c. Once every 8 microseconds.
    d. Once every 11 microseconds.
    e. Once every 15 microseconds.

Commented [TKYC27]: d. This is quite easy, just take 128 / 12000000. Catch is students either don't know how increment is related to prescalar, or they blindly use 16000000 instead.

20. Which statement best describes the output on pin PD6?

    a. PD6 remains permanently HIGH.
    b. PD6 remains permanently LOW.
    c. PD6 spends an equal amount of time HIGH as it does LOW.
    d. PD6 spends more time LOW than HIGH.
    e. PD6 spends more time HIGH than LOW.

Commented [TKYC28]: [Corrected] Answer is (a). In CTC mode, the TCNTx register will be cleared after a match. Since OCRA2 is lower than OCRB2 in this case, the `TIMER2_COMPB_vect` will never trigger, leaving the LED permanently high.

~~~ End of Questions ~~~

APPENDIX

EICRA and EIMSK layout

Name: EICRA
Offset: 0x69
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					ISC11	ISC10	ISC01	ISC00
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:2 – ISC1n: Interrupt Sense Control 1 [n = 1:0]
The External Interrupt 1 is activated by the external pin INT1 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT1 pin that activate the interrupt are defined in the table below. The value on the INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Value	Description
00	The low level of INT1 generates an interrupt request.
01	Any logical change on INT1 generates an interrupt request.
10	The falling edge of INT1 generates an interrupt request.
11	The rising edge of INT1 generates an interrupt request.

Bits 1:0 – ISC0n: Interrupt Sense Control 0 [n = 1:0]
The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in table below. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Value	Description
00	The low level of INT0 generates an interrupt request.
01	Any logical change on INT0 generates an interrupt request.
10	The falling edge of INT0 generates an interrupt request.
11	The rising edge of INT0 generates an interrupt request.

Name: EIMSK
Offset: 0x3D
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x1D

Bit	7	6	5	4	3	2	1	0
							INT1	INT0
Access							R/W	R/W
Reset							0	0

Configuration registers for Timer 2.

Name: TCCR2A
Offset: 0xB0
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	COM2A1	COM2A0	COM2B1	COM2B0			WGM21	WGM20
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Table 22-3. Compare Output Mode, non-PWM

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	Toggle OC2A on Compare Match.
1	0	Clear OC2A on Compare Match.
1	1	Set OC2A on Compare Match .

Table 22-9. Waveform Generation Mode Bit Description

Mode	WGM22	WGM21	WGM20	Timer/Counter Mode of Operation	TOP	Update of OCR0x at	TOV Flag Set on[1]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	-	-	-
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	-	-	-
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Bit	7	6	5	4	3	2	1	0
	FOC2A	FOC2B			WGM22	CS2[2:0]		
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

CA22	CA21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0		1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)