XI'AN JIAOTONG-LIVERPOOL UNIVERSITY
西 交 利 物 浦 大 学

REMOTE OPEN BOOK EXAM
ANSWER SUBMISSION
COVER SHEET

Name	Lu	Kai-Yu
Student ID Number	1614649	
Programme	Computer Science and Technology	
Module Title	Embedded Computer Systems	
Module Code	EEE 310	
Module Examiner	Dr. Suneel Kommuri	

By uploading or submitting the answers of this Remote Open Book Exam , I certify the following:

- I will act fairly to my classmates and teachers by completing all of my academic work with integrity. This means that I will respect the standards and instructions set by the Module Leader and the University, be responsible for the consequences of my choices, honestly represent my knowledge and abilities, and be a community member that others can trust to do the right thing even when no one is watching. I will always put learning before grades, and integrity before performance.
- I have read and understood the definitions of collusion, copying, plagiarism, and dishonest use of data as outlined in the Academic Integrity Policy, and cheating behaviors in the Regulations for the Conduct of Examinations of Xi'an Jiaotong-Liverpool University.
- This work is produced all on my own and can effectively represent my own knowledge and abilities.

I understand collusion, plagiarism, dishonest use of data, submission of procured work, submission of work produced and/or contributed by others are serious academic misconducts. By uploading or submitting the answers with this statement, I acknowledge that I will be subject to disciplinary action if I am found to have committed such acts.

Signature ……Kai-Yu Lu…… Date ……*2020.06.18*……

Question 1

a) Pipelining can improve the overall throughput of instructions.
Solutions for preventing pipeline hazards: Explicit pipeline, interlocks (forwarding), out-of-order execution, delayed branch, speculative execution.

b) Computed rate: 10 MHz.
Each output requires 28 multiplications and 27 additions (28 - 1 additions).
Number of arithmetic operations required per second: 550000000 arithmetic operations

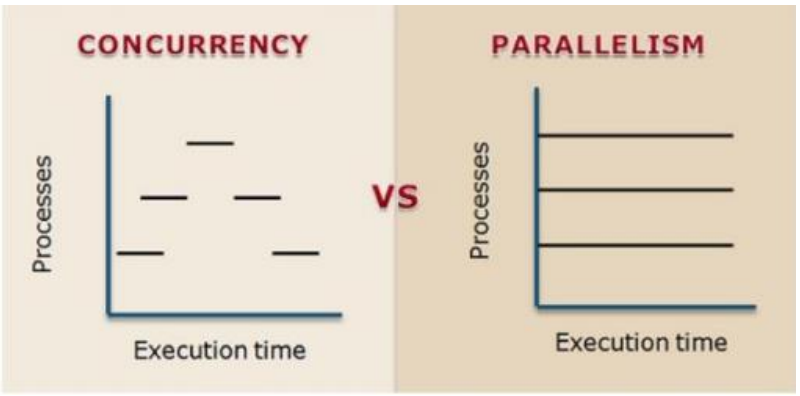
$$number\ of\ arithmetic\ operations = (multiplications + additions) * samples\ rate$$

which is $(28 + 27) * 10^7 = 550000000\ arithmetic\ operations/sec$

c) Definition of parallel: A program is parallel if different parts of the program physically execute simultaneously on distinct hardware. Parallelism means that tasks literally run at the same time. Parallelism is about doing lots of things at once. Parallel focuses on the actions executing simultaneously.

Definition of concurrent: A computer program is concurrent if different parts of the program conceptually execute at the same time (concurrently) but not necessarily simultaneously. Concurrency means that certain tasks can be started, run and completed within overlapping time periods. It does not mean that the multiple tasks will be necessarily running at the same instant. It does not completely finish one task before it begins the next. Concurrency is about dealing with lots of things at once. Concurrency focuses on the in progress at the same time.

The following figure can illustrate concurrent and parallel:



C statement for parallel:

```
/*
        Kai-Yu Lu - 1614649
        Program Description:
        This programe is to show parallel in C statement.
*/
#include <stdio.h>
double x = 3;
double xSquared, xCubed;
in main()
{
        xSquared = x * x;
        xCubed = x * x * x;
}
```

xSquared and xCubed are independent and executed in parallel. Just need to know x, then xSquared and xCubed will be known at the same time.

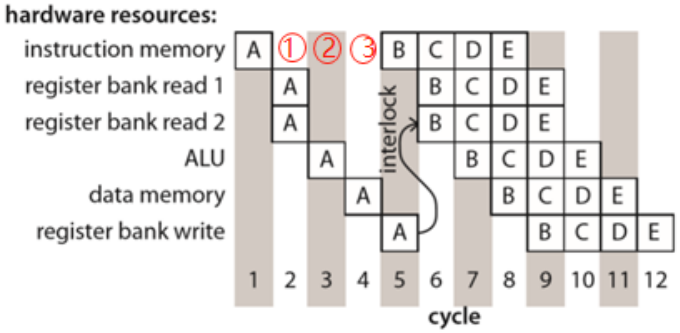
C statement for concurrent:

```
/*
        Kai-Yu Lu - 1614649
        Program Description:
        This programe is to show parallel in C concurrency.
*/
#include <stdio.h>
double x = 3;
double y = 4;
double xSquared, xCubed, ySquared, yCubed;
in main()
{
        xSquared = x * x;
        xCubed = x * x * x;

        ySquared = y * y;
        yCubed = y * y * y;
}
```

We must know what is x and y at firstly, then xSquared and ySquared will known at the same time. However, xCubed will be found after obtaining xSquared, which is same for yCubed.

d) Revised table:



There are 3 cycles are lost to the pipeline bubble. In the table, one red circle represents one cycle.

Question 2

a) It is False that the value of ``x'' passed to foo will always be 4. There is an "x" outside the main() function and this "x" is a global value. The global value can be accessed anywhere in the program. In the main(), foo(x) is executed at the beginning, then "x" is modified to 1. In this situation, the x = 4 has been passed into foo(x) before modification in "x". However, in next execution round, "x" will be 1 rather than 4, because the memory location for "x" is allocated when the program is loaded. Since in an embedded program we do not reload then if "x" is overwritten it will contain the wrong value the next time the program runs. Therefore, it is better to initialize the global variables in main(). Therefore, the statement in the question is False (x will not always be 4).

b) Procedure:

1. Set selection: The s bits encoding the set are extracted from address a and used as an index to select the corresponding cache set
2. Line matching: Check whether copy of w is present in the unique cache line. This can be done by checking valid and tag bits for that cache line.
3. Word selection: If the word present in cache block, use b bits of address a encoding the words position within the block to read that data word

Cache miss, the word w request from next level memory hierarchy

c) Memory protection units play take a significant rule since it can prevent one task disrupting with each other when multiple tasks are being executed simultaneously. It is proved to be significant in embedded applications permitting the downloading of third-party software. Additionally, protection against software bugs in security-critical applications can be supported by memory protection units. The working mechanism of memory protection unit is to assigned every address space to individual task so that a segmentation fault (or some other exception) will occur when a task attempt to access memory outside its address space

Question 3:

a) A Pulse Width Modulation (PWM) Signal is a method for generating an analog signal using a digital source. Additionally, it can deliver a variable amount of power efficiently to control external hardware devices. There are two components that defined for the behavior of a PWM signal, which are duty cycle and frequency. The duty cycle describes the amount of time the signal is in a high (ON) state as a percentage of the total time as it takes to complete one cycle. The frequency determines how fast the PWM completes a cycle and how fast it switches between high (ON) and low (OFF) states. By circularly switching a digital signal on and off at a fast rate and with a certain duty cycle, the output will appear to behave like a constant voltage analog signal when providing power to devices. Below illustrate three scenarios of duty cycles (25%, 50%, 75%) Take an electric motor as an example. If you want to speed up the motor, you can modify the PWM output to a higher duty cycle. The higher the frequency of high pulses and higher average voltage will result in faster spin of the motor.

25% Duty Cycle

50% Duty Cycle

75% Duty Cycle

Two examples using pulse width modulation: electric motor and setting the period of the on-chip timer/counter that provides the modulating square wave

b) Serial Interface: The number of pins on the processor integrated circuit is limited
Parallel Interface: Parallel interface uses multiple lines to simultaneously send bits
Parallel Interface delivers higher performance because more wires are used for the interconnection.

c) When an interrupt is asserted by changing the voltage on a pin, it will cause response may be edge triggered or level triggered. Edge triggered changes the voltage for only a short-time while level triggered holds voltage on the line until gets acknowledgment. Therefore, the next state of voltage would be difficult to understand and will many catastrophic failures will occur due to unexpected behaviors.
Function of finite state machine (FSM): Because the next state will be uncertain when an interrupt is asserted by changing the voltage on a pin, FSM has a finite number of states and the flow of logic execution depends on the flow or transition between states. The transition from one state to another state will depend on the current state the machine is in and on the input into a state. Therefore, the previously mentioned catastrophic failures caused by the response (edge triggered/level triggered) could be avoided.

d)

```
/*
        Kai-Yu Lu - 1614649

        Program Description:
        Create a C program for the ARM CortexTM – M3 to use
        the SysTick timer to invoke a system-clock ISR with a jiffy
        interval of 10 ms that records the time since start in a 32-bit int.
        Suppose that for every 10 ms, compiler would like to count down
        from some initial count until the count reaches zero, and then stop
        counting down.

*/

volatile unit timerCount = 0;
void ISR(void){
```

```
        if (timerCount != 0){
                timeCount --;
        }
        SysTickEnable();
}

SysTickPeriodSet(SysCtlClockGet() / 1000);

int main(void){
        //initialization code
        timeCount = 10;
        SysTickIntRegister(&ISR);
        SysTickIntEnable();
        while(timerCount !=0){
                // code to run for 10 seconds
        }
}
```

There are 2^{32} -1 long this program run before your clock overflows