University of Victoria - Final Exam

December 2023

Student's last name:	
Student's first name:	
Student number:	

Course title:	*Microprocessor-Based Systems*
Course number:	*ECE 355*
Section(s):	*A01*
CRN:	*11024*
Instructor:	*Daler Rakhmatov*
Duration:	*3 hours*

This exam has a total of:

- 7 pages (including this cover page)
- 0 additional pages of separate handouts

Count the number of pages you have and inform the invigilator if any are missing.

This exam is to be answered:

- In booklets provided

Students are allowed to use the following materials:

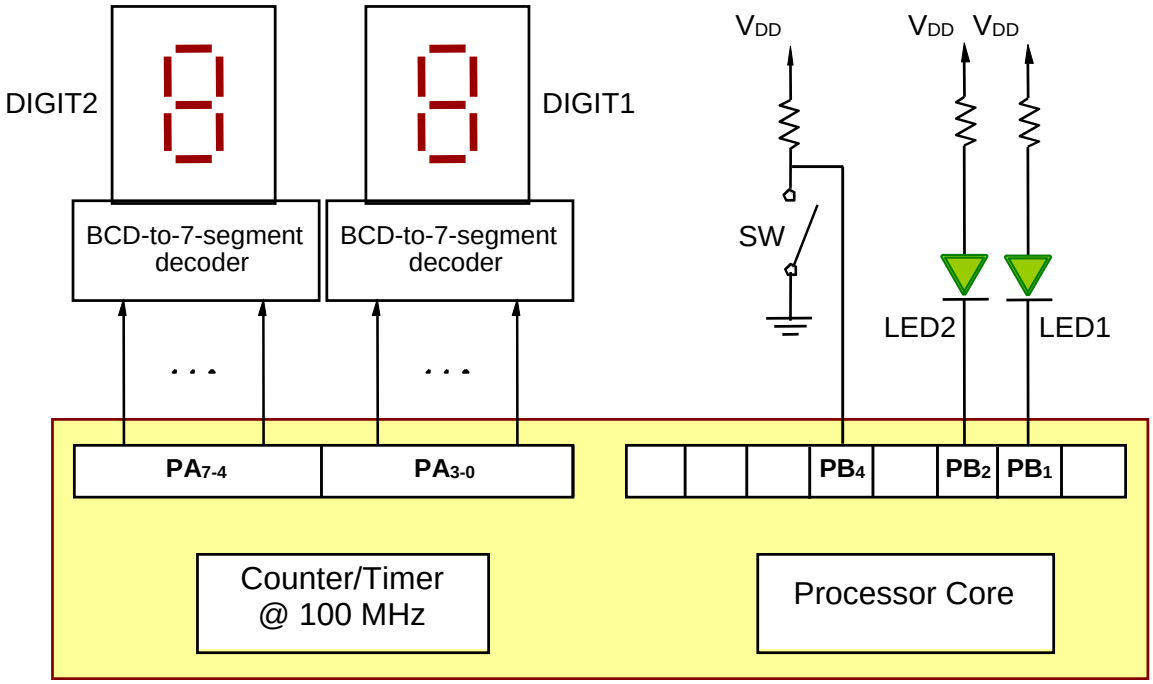
- Anything printed or handwritten, stand-alone calculators

1. *[15 points]* The textbook's microcontroller is used in a system shown below, and it is responsible for two tasks: 1) alternating between **LED1** and **LED2** being on, whenever the **SW** key is hit (i.e., pressed and then released), and 2) decrementing either **DIGIT1** (when **LED1** is on) or **DIGIT2** (when **LED2** is on) every second. Write the corresponding C program, assuming that the **first task** is the main program, and the **second task** is an ISR whose address is stored at location **0x20**. Also, assume that bit **6** of the processor status register (i.e., **PSR[6]**) is the processor's interrupt-enable bit, and **Ports A** and **B** are always ready to be written by the processor. Initially, **LED1** is on, **LED2** is off, and both **DIGIT1** and **DIGIT2** show **0**.

• *Main Program*: Whenever **PB_4** first becomes 0 and then 1 again, **LED1** and **LED2** must swap their states: if **LED1** is on and **LED2** is off, then **LED1** becomes off and **LED2** becomes on, and vice versa. **Note: LED1** and **LED2** should never be both on, or both off.

• *ISR*: The 100-MHz Counter/Timer must be configured to generate interrupts every second. Its ISR must decrement **DIGIT1** if **LED1** is on, or decrement **DIGIT2** if **LED2** is on. (**Note:** decrementing **0** gives **9**.)

You do **NOT** need to write down **#define** statements.



2. *[20 points]* Consider a byte-addressable computer with 4-KB main memory and 128-byte cache having **eight blocks**, where each block consists of **four 32-bit words**. Assume that the CPU reads 32-bit words from the following sequence of hexadecimal addresses:

03C FF4 050 070 078 0F0 FF4 03C 070 078

Show the cache contents (e.g., **[000]** = contents stored at address **000**) at the end of this sequence (10 addresses) and calculate the corresponding miss rate given that:

(a) Cache is direct-mapped.
(b) Cache is 2-way set-associative (2 blocks per set) with LRU replacement.
(c) Cache is 4-way set-associative (4 blocks per set) with LRU replacement.
(d) Cache is fully-associative with LRU replacement.

3. *[15 points]* Consider a C code fragment below, processing some matrix **`float X[N][N]`** (stored row by row, i.e., in the row-major order), where **`N = 512`**:

```
float y, neg = 0;
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
        if (X[i][j] < 0) neg++;      /* count negative elements */
    }
}
y = neg/(N*N);
for (i = 0; i < N; i++) {
    X[i][i] = X[i][i] + y;           /* add to diagonal */
}
```

Determine the **x**-related page fault rate in the following three cases: 1) the main memory uses **1-KB** paging with 4 pages allocated for **x**, 2) the main memory uses **2-KB** paging with 2 pages allocated for **x**, and 3) the main memory uses **4-KB** paging with 1 page allocated for **x**. Initially, no part of **x** is in the main memory. **Note:** **`float`** = 4 bytes.

4. *[5 points]* The table below specifies a set of four independent pre-emptive tasks to be executed by a single processor. Show the task schedule using the **Earliest Deadline First** (EDF) priority assignment. If some tasks happen to have the same EDF priority, break such ties using **Rate Monotonic** (RM) prioritization.

Task $\mathbf{T_i}$	Period $\mathbf{P_i}$	WCET $\mathbf{C_i}$	Deadline $\mathbf{D_i}$	Initial Delay ϕ_i
T1	30	10	30	0
T2	40	10	40	0
T3	60	10	60	0
T4	120	20	120	0

5. *[5 points]* Consider a pipelined datapath consisting of five stages:

- **F** – fetch the instruction from the memory,
- **D** – decode the instruction and read the source register(s),
- **C** – execute the ALU operation specified by the instruction,
- **M** – execute the memory operation specified by the instruction,
- **W** – write the result in the destination register.

Identify data hazards in the code below and insert **NOP** instructions where necessary:

```
MOV   (R0), R1          // R1 = MEMORY[R0]
ADD   #4, R0, R0        // R0 = R0 + 4
MOV   (R1), R2          // R2 = MEMORY[R1]
ADD   #4, R1, R1        // R1 = R1 + 4
SUB   R3, R0, R3        // R3 = R3 - R0
MOV   R4, (R2)          // MEMORY[R2] = R4
SUB   R4, R1, R4        // R4 = R4 - R1
MOV   R3, (R1)          // MEMORY[R1] = R3
ADD   #4, R4, R1        // R1 = R4 + 4
ADD   #4, R3, R0        // R0 = R3 + 4
```

6. *[10 points]*

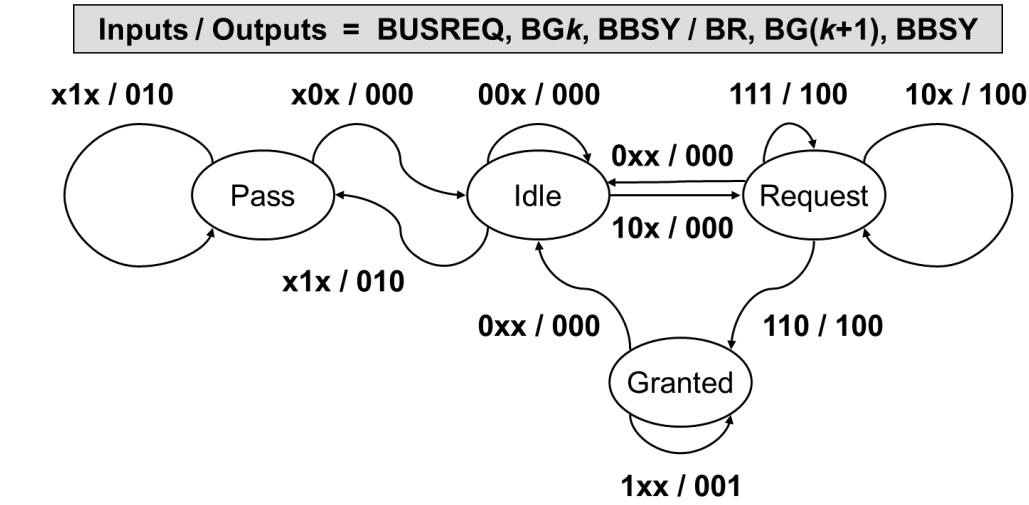
(a) Show **decimal** number **–21.25** in the 32-bit IEEE-754 floating-point representation.

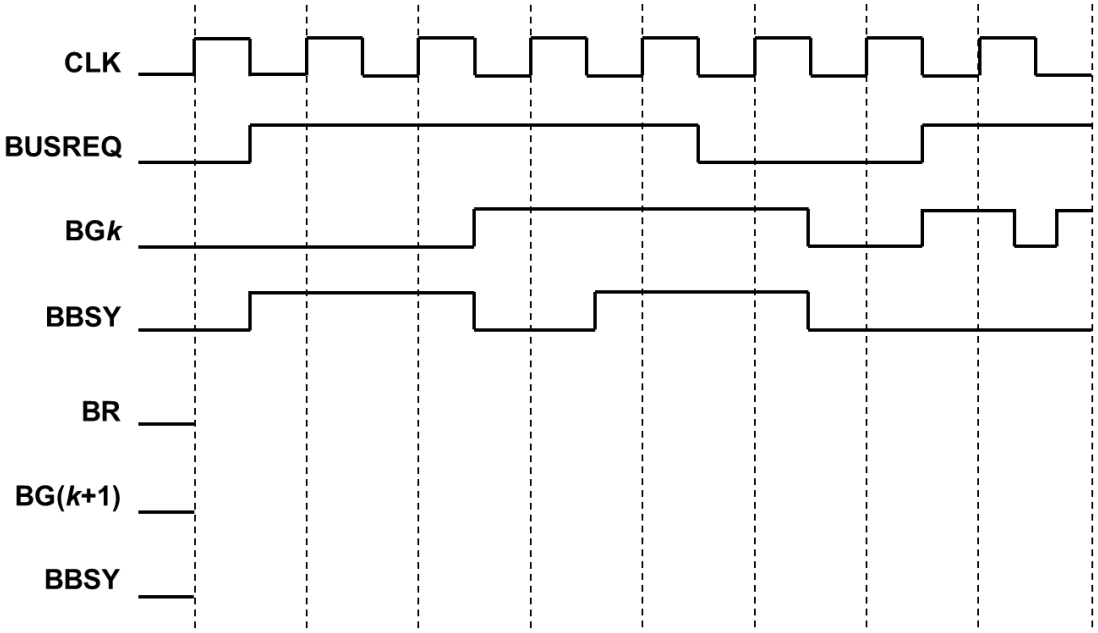
(b) Given two 32-bit **IEEE-754** floating-point numbers **X** and **Y** below, calculate (in the binary format) **Z=X+Y**. Then, convert your IEEE-754 result **Z** to the decimal format.
X = **1 01111011 01100000000000000000000**,
Y = **1 01111100 11010000000000000000000**.

7. *[5 points]* Suppose some **Moore-type** subordinate FSM has 2 inputs **SEL** and **READ** and 2 outputs **READY** and **LAST**. Show the state diagram of this FSM that implements the following protocol: 1) Initially, the FSM outputs **READY = LAST = 0** and waits for **SEL = 1** and **READ = 1**; 2) Once **SEL = 1** and **READ = 1** are received, the FSM outputs **READY = 0** and **LAST = 0** for one clock cycle; 3) Then, the FSM outputs **READY = 1** and **LAST = 0** for two clock cycles; 4) Once these two clock cycles have elapsed, the FSM outputs **READY = 1** and **LAST = 1** and waits for **SEL = 0**; 5) Once **SEL = 0** is received, the FSM goes back to step 1).

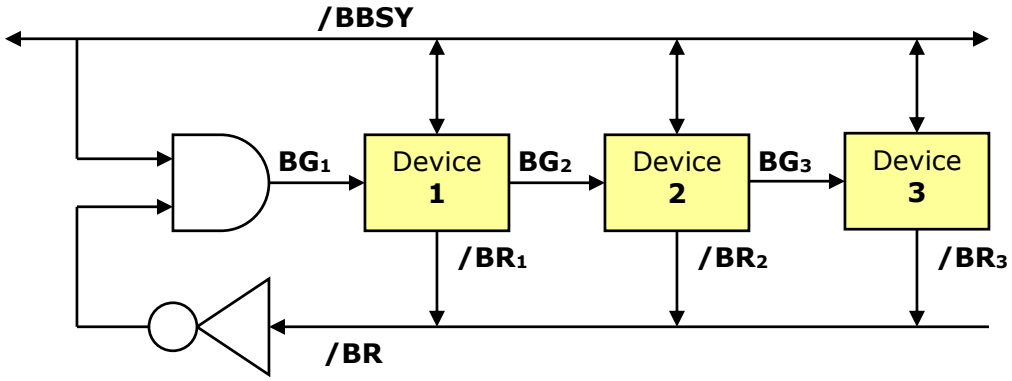
Note: If **SEL** becomes **0** during steps 2), 3), or 4), the FSM goes back to step 1). The value of **READ** is important only during step 1).

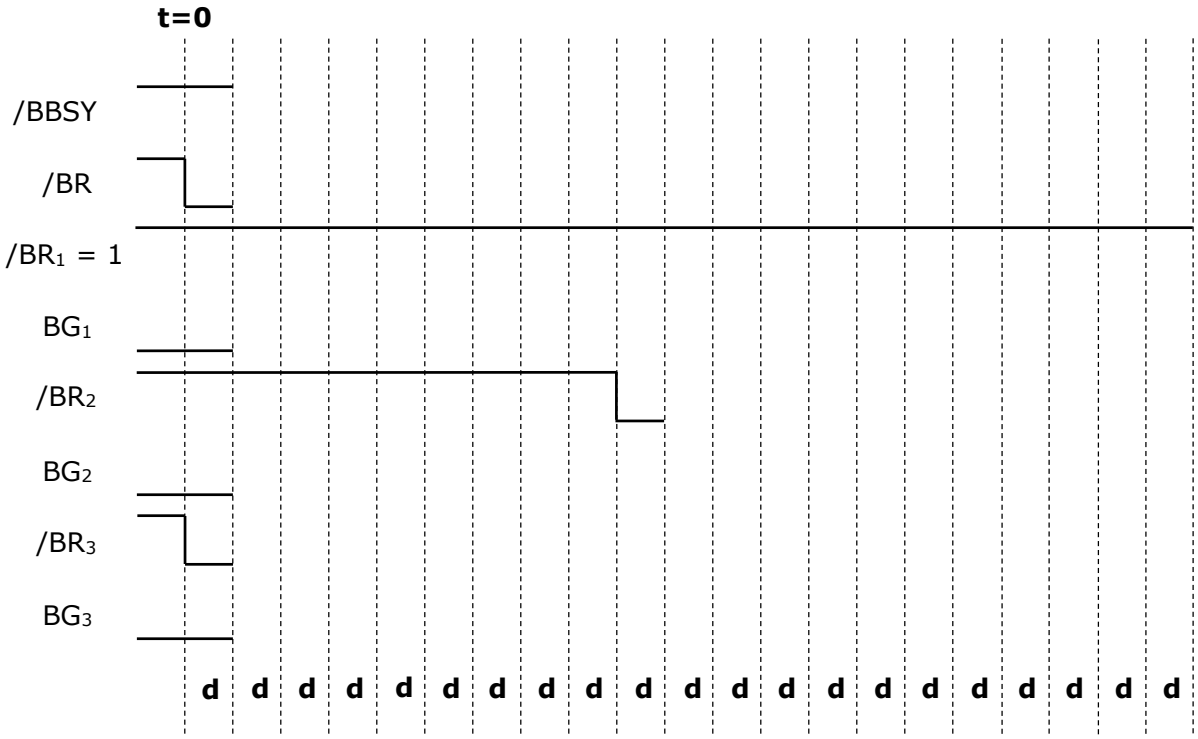
8. *[5 points]* Consider the Mealy FSM state diagram of some daisy-chain device as shown below, where **x** represents **don't-care**. Given the input waveforms shown below, draw the corresponding output waveforms, assuming that the FSM is initially in state **Idle**.





9. *[10 points]* Consider the daisy-chain arbitration scheme shown below. Assume that the input-to-output signal propagation delays are the same and equal to **d** for all three devices, the inverter, and the **AND** gate. Requesting device **x** is able to start using the bus (making **/BRx = 1** and **/BBSY = 0**) only when it receives a **0-1** transition on its bus-grant input **BGx** and detects that the bus is not currently busy (i.e., **/BBSY = 1**). Device **x** lets the bus-grant propagate through only when it is neither requesting nor using the bus. Assume that any of the three devices will use the bus (when granted) for **3d** time units. Complete the timing diagram shown on the next page.

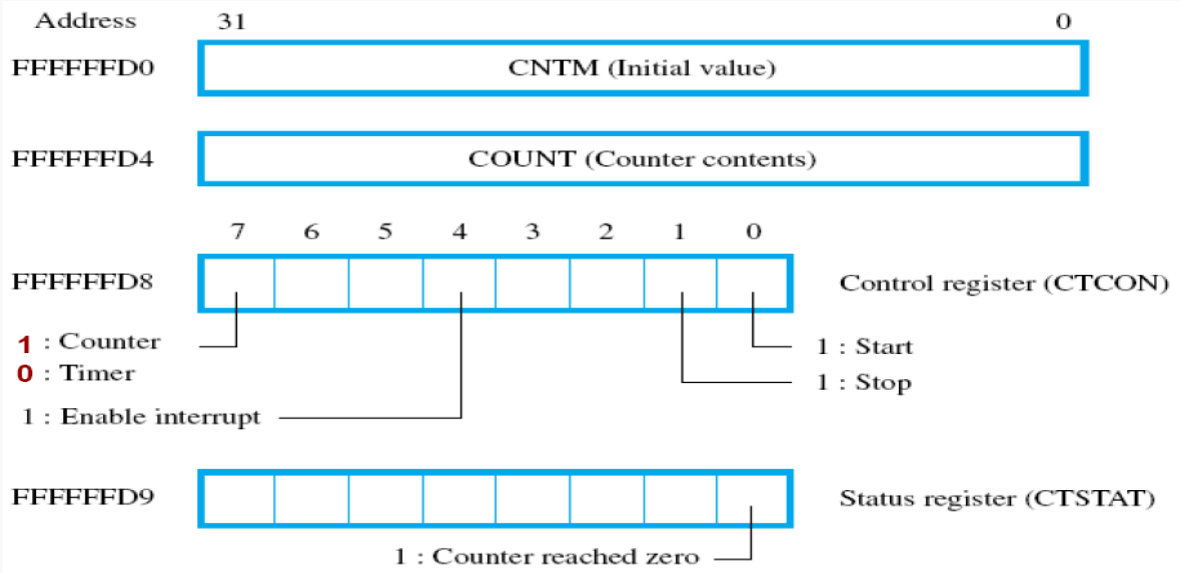




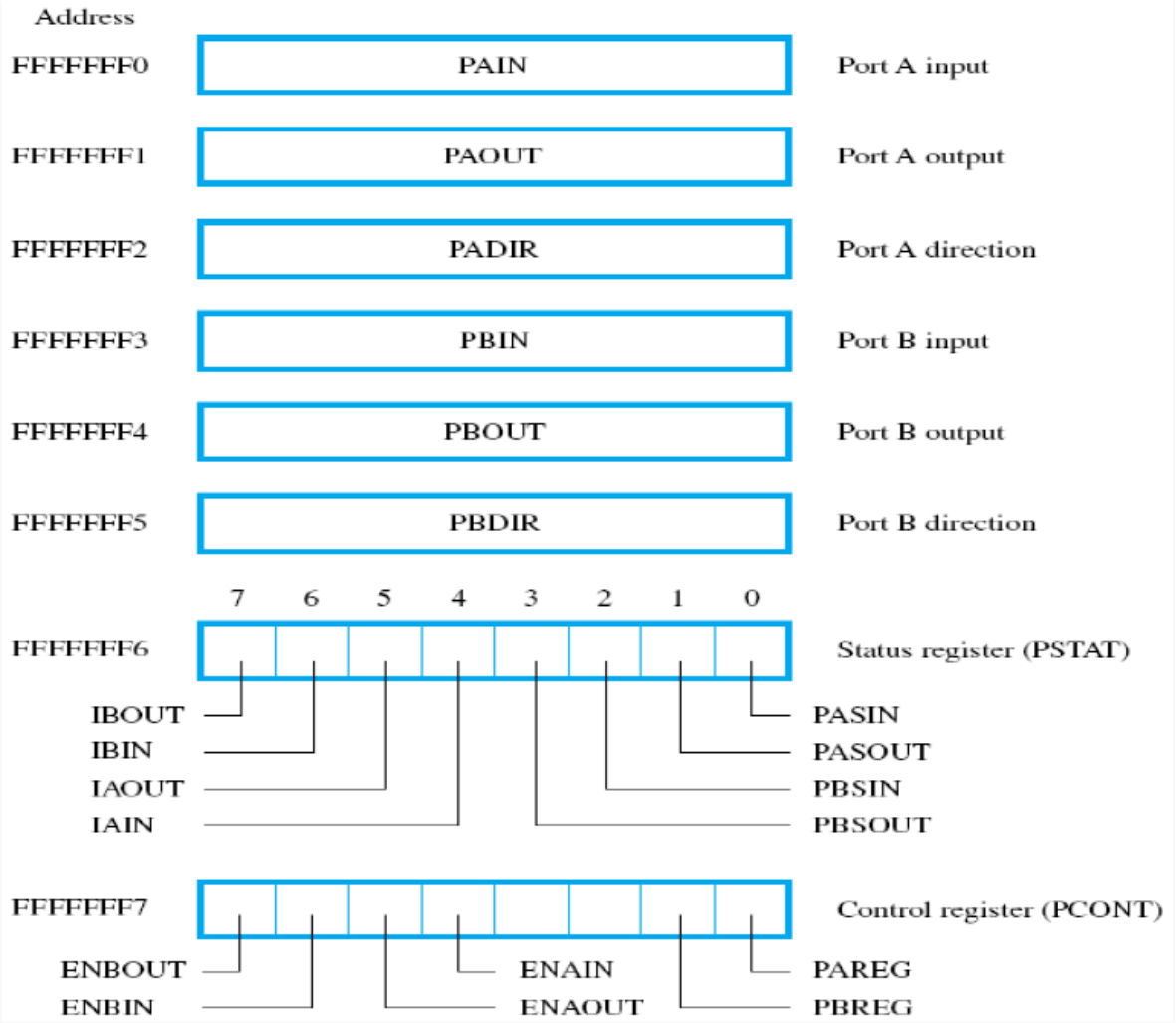
***** Please INSERT your ENTIRE EXAM PAPER into your BOOKLET *****

***** Please INSERT your ENTIRE EXAM PAPER into your BOOKLET *****

Counter/Timer Registers



Parallel Port Registers



END