**EASTERN MEDITERRANEAN UNIVERSITY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CMSE 443**

**Real-Time Systems Design**

**Quiz (5 points)**

**2019/20 Spring Semester**

**June 12, 2020, Friday, 14.30**

Assoc. Prof. Dr. Alexander CHEFRANOV, e-mail: [Alexander.chefranov@emu.edu.tr](mailto:Alexander.chefranov@emu.edu.tr)

Time: 110 minutes

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **QUESTION** | **T1** | **T2** | **T3** | **T4** | **T5** | **T6** | **T7** | **T8** | **T9** | **Total** |
| **Out of 100%** | 4 | 9 | 9 | 4 | 13 | 10 | 17 | 17 | 17 | 100%=5 points |
| **Grade** |  |  |  |  |  |  |  |  |  |  |

**Please, consider the instructions below. The tasks are in pages 2-4.**

* Your answers shall be hand-written
* On each page, at the top, write a header: “CMSE443 Quiz 12.06.2020”, followed by your Name, Surname, Student ID, page number
* There are **9** tasks in total. Try answering each task. There are 4 pages in total.
* You may rewrite the text of the task in your paper, or not. It is up to you. But at the beginning of an answer, write “Task <i> answer:” substituting <i> by a particular task number
* Open book, open notes, work yourself
* Copies are not allowed and will be zero graded
* In **30 minutes after the exam finishing**, you shall do **all** the following.

1. Make photo of each your page so that its full content, including the header, is in the image, and clearly readable.

2. Then, the images shall be assembled in the page number order into a pdf file named “CMSE443 Quiz12062020 <stid>.pdf” with <stid> substituted by your student identification number.

3. **Finally,** e-mail it to [Alexander.chefranov@emu.edu.tr](mailto:Alexander.chefranov@emu.edu.tr)

* Good Luck!

**Task 1. (4 points)** What deterministic algorithm is? What nondeterministic algorithm is? Why Real-Time system algorithms may behave nondeterministically?

Deterministic means that for any allowed input, output is determined uniquely

Nondeterministic algorithm, allows for a particular input having more than one possible output

In RTS, algorithms may behave indeterministically, e,g., when the data used are not properly initialized, or the controlled object enters a not expected state

**Task 2. (9 points)** Consider 1-address machine, with one register available, instruction set below

Load R,mem// load register R from memory cell, mem

Sto R, mem// store content of register, R, to memory cell, mem

Add R,mem// add content of register, R, and content of memory cell, mem; result

//is saved in R

Sub R,mem// subtract from register, R, content of memory cell, mem; result is

//saved in R

Mul R, mem// multiply register, R, content with memory cell, mem, content;

//result is saved in R

Div R, mem //saved in R; division is integer-valued,, e.g., ½=0; 3/2=1; 4/2=2

Assume that memory cell allocated to variable, x, is x.

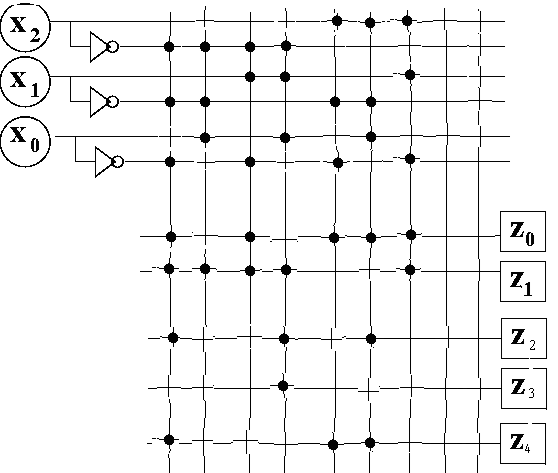
For the expression

1. Write the code to implement it (4 points)
2. Load r, z
3. Div r, y
4. Sto r, t1
5. Load r. w
6. Sub r, z
7. Mul r, t1
8. Add r, y
9. Sto r, y

1. Trace the code assuming w=2, x=3, y=3, z=2, R=0: show state of the memory cells allocated for the variables and state of R initially and after each instruction completion. (5 points)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Instruction# | y | z | w | r | T1 |
| 3 | 2 | 2 | 0 |  |
|  |  |  |  | 2 |  |
|  |  |  |  | 0 |  |
|  |  |  |  |  | 0 |
|  |  |  |  | 2 |  |
|  |  |  |  | 0 |  |
|  |  |  |  | 0 |  |
|  |  |  |  | 3 |  |
|  | 3 |  |  |  |  |

**Task 3. (9 points)** Use PLA to create a ROM with 7 memory cells containing the following numbers in the given order: 23, 2, 3,14, 17, 21, 3~~, 9~~. For each number, specify its address in your ROM. Each memory cell shall have one and the same number of bits, and this number of bits shall be minimal possible to accommodate the specified seven numbers. What is the number of bits of each memory cell? Why? What is the number of address bits? Why?



**Task 4. (4 points)**. How CPU knows what interruption service routine shall be invoked when an interruption signal number 5 is raised? Give necessary explanations.

Interruption vectors are kept in the 1st kilobyte of the memory, and i-th vector is kept in 4 bytes starting from the address number 4\*I, i=0,255. Interruption vector keeps address of the interruption service routine For interruption signal 5, the interruption vector resides in the bytes 20-23.

**Task 5. (13 points).** Consider the code and diagram below

|  |  |
| --- | --- |
| Driver{ while(1){  if(data\_for\_I/O){  prepare(command);  V(busy); P(done);}  }  }  Controller{while(1){  P(busy);  exec(command);  V(done);  }  } |  |

Explain, how Controller and Driver interact. What synchronization mechanism they use for it? Explain what particular functions of that mechanism are used and what for? Give definition of these functions. Draw a sequence diagram to illustrate their interaction for two iterations of each of them. Provide initializations necessary to provide a scenario you show.

Controller and Driver interact via synchronization mechanism. They use binary semaphores for synchronization. Synchronization functions P() and V() are used. P(S), wait semaphore function, if S=1 (open), sets S=0 (closes it), and terminates. If S=0 (closed), the processes waits in the function for S opening, then closes it, and terminates. V(S) sets S=1 (open) and terminates. The semaphores, are initialized: busy=0 (closed), done=0 (closed).

|  |  |  |
| --- | --- | --- |
| Time | Driver | Device |
|  | Prepare(command) | Wait at busy: P(busy) |
|  | Command prepared: signal device: V(busy); Wait at done for the command termination: P(done) | Proceed to command execution |
|  | Proceed to the next command preparing | Signal that the command is finished: V(done); wait for the next command: P(busy); |

**Task 6 (10 points).** Calculate processor utilization and hyper-period for the following task set:

|  |  |  |
| --- | --- | --- |
| **Task#** | **E** | **P** |
| **1** | **3** | **5** |
| **2** | **4** | **6** |
| **3** | **5** | **7** |
| **4** | **6** | **8** |

Utilization=3/5+4/6+5/7+6/8=0.6+0.67+0.71+0.75=2.73 (273%)

Hyperperiod=lcm(5,6,7,8=5\*6\*7\*4=840

**Task 7. (17 points).** Build a Rate-Monotonic schedule for tasks:

|  |  |  |
| --- | --- | --- |
| Task# | E | P |
| 1 | 5 | 10 |
| 2 | 2 | 5 |
| 3 | 2 | 8 |

What are the utilization and hyperperiod for this set of tasks? Are the deadlines met by the RM schedule? Explain why they are met (not met)?

Utilization=5/10+2/5+2/8=0.5+0.4+0.25=1.15 = 115%

Hyperperiod=lcm(10,5,8)=5\*8=40

2-1

2-1

3-1

 3-1

1-1

 2-2

 2-2

 1-1

3-2

3-2

 2-3

 2-3

1-1

 1-1

 1-1

2-4

2-4

3-3

 3-3

 1-2

 2-5

 2-5

 1-2

 1-2

3-4

 2-6

2-6

 3-4

 1-2

 1-2

 2-7

 2-7

3-5

 3-5

 1-2

 2-8

 2-8

 1-2

 1-2

 1-3

Since utilization is greater than 1, the lowest priority task 1 violates its deadlines, and not all its releases are completed inside the hyperperiod. In the diagram, m-n means task number m, release number n.

**Task 8 (17 points)**. Assume, three parallel processes, A1, A2, A3, infinitely run. Process A2 waits for a command from A1, executes it and signals to A3 about its completion. A1 after signaling to A2 waits for completion of the work by A3. Write a pseudo-code using binary semaphores to provide necessary synchronization of the processes A1. A2, A3. Define necessary data structures. Show initial settings of the semaphores you use.

Semapher S12=0, S23=0; S31=0;// 0 means closed

|  |  |  |
| --- | --- | --- |
| A1 | A2 | A3 |
| Loop: begin | Loop: begin | Loop: begin |
| Exec; | Wait(S12); | Wait(S23); |
| Signal(S12); | Exec; | Exec; |
| Wait(S31) | Signal(S23) | Signal(S31) |
| end | end | end |

**Task 9 (17 points)**. Assume that a system has 4 processes and resources of one type: memory (total available number of memory blocks is 10). Processes’ resources required and maximal requirements are as follows:

|  |  |  |
| --- | --- | --- |
| Process | Memory required | Memory maximal required |
| 1 | 3 | 7 |
| 2 | 1 | 4 |
| 3 | 1 | 5 |
| 4 | 2 | 3 |

Currently, each process has no any memory unit allocated to it. Use the Banker algorithm to decide on safety of granting required resources. Show all the steps of the algorithm you make to come to your decision.

If granting all the requirements, the total number of memory units granted is S=3+1+1+2=7, and the number of the resources left, r=Total-S=10-7=3.

The maximal resource required for the processes are: P1=>4; P2=>3; P3=>4; P4=>1

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Memory required | Memory maximal required | Max needed |
| 1 | 3 | 7 | 4 |
| 2 | 1 | 4 | 3 |
| 3 | 1 | 5 | 4 |
| 4 | 2 | 3 | 1 |
|  |  | Total left, r: | 3 |

If the maximal left requirements are actually needed, it is possible first granting to P2 3 units, and r=r-3=0. When P2 finishes, the r=r+4=4. Then P1 can be granted: r=r-4=0. When P1 finishes, it releases its resources: r=r+7=7. Then P3 can be granted 4 units: r=r-4=3. And. concurrently, P4 can be granted 1 unit: r=r-1=2. When P3 finishes, r=r+5=7. When P4 terminates, r=r+3=10, equal to the total resource number available in the system. The state is safe. The safe sequence is (P2, P1, P4, P3).