Politecnico di Milano
SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

Advanced Operating Systems
A.A. 2019-2020 – Exam date: July, 21th 2020

Prof. William FORNACIARI

Surname (readable)...................................... Name (readable)...

Matr... Signature...

Q1	Q2	TOT

NOTES

It is forbidden to refer to texts or notes of any kind as well as interact with their neighbors. Anyone found in possession of documents relating to the course, although not directly relevant to the subject of the examination will cancel the test. It is not allowed to leave during the first half hour, the task must still be returned, even if it is withdrawn. The presence of the writing (not delivered) implies the renunciation of any previous ratings.

Question 1 (12 points)

A timer peripheral in a microcontroller exposes the following registers:

Bit access modes:

- **u** = undefined, writing has no effect, reading returns zero
- **w** = always reads as zero, writing 1 causes the corresponding action
- **rw** = software reading and writing is allowed
- **rc-w1** = bit is readable. Writing 0 has no effect, writing 1 clears the bit

TIMS: Address: 0xFED0, Initial value upon reset: 0x0000

7	6	5	4	3	2	1	0
IE	IF	AR	UD	ST	-	-	EN
rw	rc_w1	rw	rw	rw	u	u	rw

- IE: Interrupt enable. If set, a timer overflow/underflow will cause IF to be set and the interrupt routine void TIM_irq(); to be called.
- IF: Interrupt flag. Must be cleared before returning from the interrupt routine.
- AR: Auto restart. If set the timer will continue counting after an overflow/underflow. If cleared, upon overflowing/underflowing ST is cleared in hardware and the timer stops immediately following an overflow/underflow (at 0 if counting up, or at 255 if counting down).
- UD: Up/down. Setting the but causes the timer to count up, clearing it makes it count down.
- ST: Start. The timer is incremented/decremented every millisecond as long as this bit is set.
- EN: Enable. This bit must be set first to enable the peripheral.

TIMC: Address: 0xFED1, Initial value upon reset: 0x0000
This 8 bit register holds the timer counter value. It can be read/written at any time by software and is incremented/decremented in hardware if the timer is enabled and started.

Assuming data structures and macros to access these peripheral registers by a C/C++ program, using the syntax TIMS=1; TIMC=1; **are already available**, you are required to implement the following C++ class whose constructor initializes the timer, and whose registerCallback member function is a **nonblocking** function that will call the function passed as argument from the interrupt routine periodically with the given period (you can assume periodMilliseconds must be between 1 and 250 ms). Calling registerCallback again will replace the previous callback with the new one.

```
class Timer
{
public:
    Timer();
    void registerCallback(std::function<void ()> function, int
periodMilliseconds);
private:
    // Add methods and variables as needed
};
```

Solution

```
//timer.h

class Timer
{
public:
    Timer();
    void registerCallback(std::function<void ()> function, int periodMilliseconds);
};


//timer.cpp

static std::function<void ()> func;
static int reload;

void TIM_irq()
{
    TIMC = reload;
    TIMS |= (1<<6)  //IF
          | (1<<3); //ST

    //Call function after restarting the time, or the time spent in the function
    //will add skew to the period
    func();
}

Timer::Timer()
{
    TIMS = (1<<0)  //EN
         | (1<<7); //IE
}

void Timer::registerCallback(std::function<void ()> function, int periodMilliseconds)
{
    if(periodMilliseconds < 1 ||  periodMilliseconds > 250) return;

    //Stop the timer, so that if this function is called multiple times we avoid the
    //race condition of the timer interrupt firing while we are writing to the variables
    //func and reload that are shared between the main code and interrupt.
    //We also create a bitmask that leaves bit 6 at 0, as it’s an rc_w1 and in general
    //we don’t want to clear those bits accidentally
    TIMS &= ~((1<<3) | (1<<6));

    func = function;
    TIMC = reload = periodMilliseconds - 1;

    TIMS |= (1<<3);
}
```

Question 2 (11 points)

Given the following task set, you are asked to draw the Gantt diagram of the schedule obtained by applying each of the following algorithms on a single core processor:

- Shortest Remaing Time First (SRTF)
- Highest Response Ratio Next (HRRN)

Task	Arrival time (a)	Completion time (C)
0	0	7
1	4	2
2	7	3
3	5	4

In case of tasks with the same priority, pick the one with the lowest id number.

For each algorithm:

- Fill a table reporting arrival, start and finish time, other than computation, waiting, response and turnaround time for each task

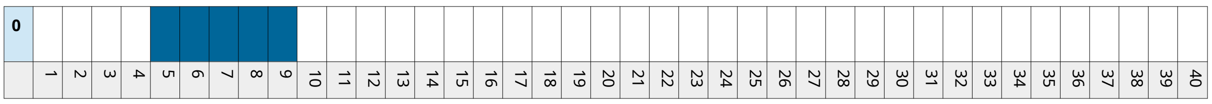
- Report the status of the scheduling queue status related to the most important time instants.

- Compute the average values of waiting time (W), response time (R) and turnaround time (Z).

Finally, make some comments on the performance shown by the different scheduling algorithms, according to the metrics computed above.

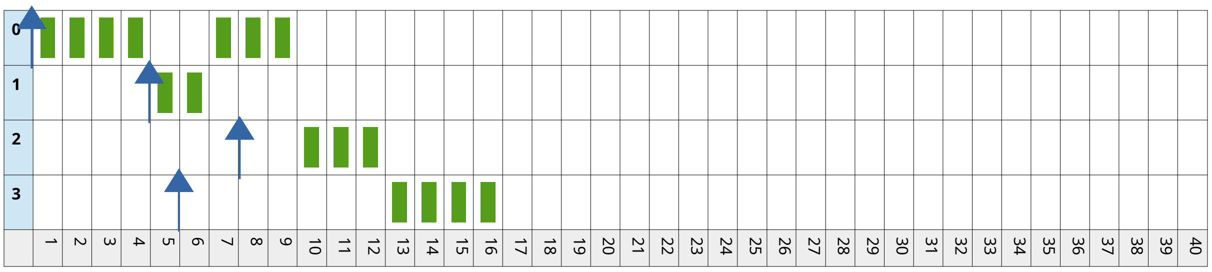
Example

*Task 0 started at **t=4** and terminated at **t=9**.*



1) Shortest Remaining Time First (SRTF)

Schedule



Task	Arrival time	Start time	Finish time	Computation time (C)	Waiting time (W)	Response time (R)	Turnaround time (Z)
0	0	0	9	7 (3)(2)	2	4	9
1	4	4	6	2 (1)	0	2	2
2	7	9	12	3	2	5	5
3	5	12	16	4	7	11	11

Status of the queue

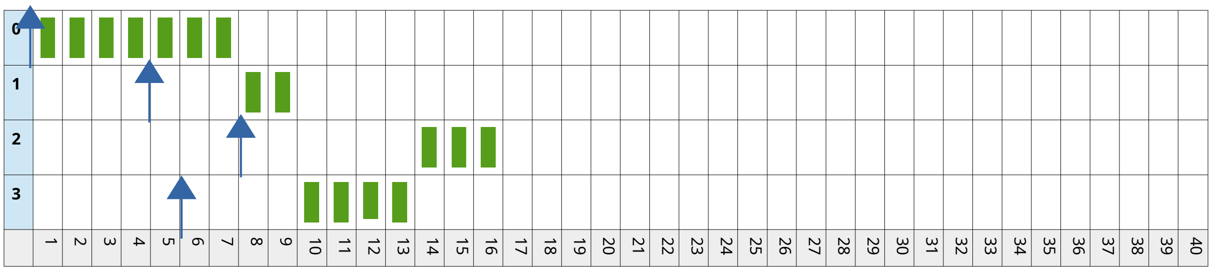
t4: $\mathbf{C_{T1}=2}$, $C_{T0}=3$
t5: $\mathbf{C_{T1}=1}$, $C_{T0}=3$, $C_{T3}=4$
t6: $\mathbf{C_{T0}=3}$, $C_{T3}=4$
t7: $\mathbf{C_{T0}=2}$, $C_{T3}=4$, $C_{T2}=3$
t9: $C_{T3}=4$, $\mathbf{C_{T2}=3}$
t12: $\mathbf{C_{T3}=4}$

W = (2+0+2+7) / 4 = 2.75
R = (4+2+5+11) / 4 = 5.5
Z = (9+2+5+11) / 4 = 6.75

2) Highest Response Ratio Next (HRRN)

Task	Arrival time (a)	Completion time (C)
0	0	7
1	4	2
2	7	3
3	5	4

Schedule



Task	Arrival time	Start time	Finish time	Computation time (C)	Waiting time (W)	Response time (R)	Turnaround time (Z)
0	0	0	7	7	0	7	7
1	4	7	9	2	3	5	5
2	7	13	16	3	6	9	9
3	5	9	13	4	4	8	8

Status of the queue

t7: $\mathbf{RR_{T1}=(3+2)/2=2.5}$, $RR_{T2}=(0+3)/3=1.0$, $RR_{T3}=(2+4)/4=1.5$
t9: $RR_{T2}=(2+3)/3=1.66$, $\mathbf{RR_{T3}=(4+4)/4=2.0}$

W = (0+3+6+4) / 4 = 3.25
R = Z = (7+5+9+8) / 4 = 7.25