Computer Architectures
Exam of 12/02/2025

B2

Please read the following notes before proceeding

1) It is recommended to save the project on the C drive during development. Once you have completed your work, you must save the project inside the exam folder on the Z drive (the same folder where you found this document). Projects saved elsewhere will not be collected and will be permanently lost.
2) The assembly subroutine developed in the first exercise must comply with the ARM Architecture Procedure Call Standard (AAPCS) standard (in terms of parameter passing, returned value, callee-saved registers).
3) If you prefer, you can begin with the second exercise. Since it requires calling the assembly subroutine from the first exercise, you can create a stub in the assembly file to proceed. For example:

```
subroutineName    PROC
      EXPORT subroutineName
      MOV r0, #10       ; example of a return value
      BX LR
      ENDP
```

4) To earn points on both exercises, it is recommended that you write at least some code for each.
5) The final project must run on the actual board.

Special characters with Italian keyboards

- Square brackets [ ]
  - Left bracket: ATL GR + [
  - Right bracket: ALT GR +]
- Curly brackets { }
  - Left bracket: LSHIFT + ALT GR + [
  - Right bracket: LSHIFT + ALT GR +]
- Hash # : ALT GR + à

Bitwise operations in C

Left shift by N bits = A<<N
Right shift by N bits = A>>N

A XOR B = A^B
A AND B = A&B (&& is the logical AND)
A OR B = A|B (|| is the logical OR)

Computer Architectures
Exam of 12/02/2025

B2

Question 1 (10 points)

Introduction

A Maclaurin series represents a function as an infinite sum of terms, which are computed from the values of the function's derivatives at zero. The Maclaurin series of the cosine is:

$$\cos x \cong \sum_{n=0}^{\infty} \frac{(-1)^n\, x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots \qquad (1)$$

In equation (1), both *x* and sin *x* are real numbers. They can be expressed as integer numbers to the detriment of precision. In particular, the argument of cosine is multiplied by 10 (i.e., with substitution $y = 10x$) and then cast to an integer; the result is multiplied by 100 and cast to an integer. We obtain

$$100 \cdot \cos y \cong \sum_{n=0}^{\infty} \frac{100 \cdot (-1)^n\, (y)^{2n}}{(2n)! \cdot 10^{2n}} = 100 - \frac{y^2}{2! \cdot 10^2} + \frac{y^4}{4! \cdot 10^4} - \frac{y^6}{6! \cdot 10^6} + \cdots \qquad (2)$$

Assignment

Write a `Maclaurin_cos` subroutine that computes the value of the Maclaurin series of the cosine at a point *x*, up to order *n*. The subroutine receives in input the integer values *y* and *n*; the returned value is 100 times the value of the Maclaurin series of cos *y*.

Suggestion: according to equation (2), the element of order $n = 0$ is $t_0 = 100$. Then, any new element t_i can be computed based on the previous element t_{i-1} as follows:

$$t_i = \frac{-t_{i-1} \cdot y^2}{(2i-1) \cdot (2i) \cdot 100} \qquad (3)$$

Example: the input parameters are $y = 25$ and $n = 4$.
$t_0 = 100$
$t_1 = (-100 \cdot 25^2) / (1 \cdot 2 \cdot 100) = -62500 / 200 = -312$
$t_2 = (312 \cdot 25^2) / (3 \cdot 4 \cdot 100) = 195000 / 1200 = 162$
$t_3 = (-162 \cdot 25^2) / (5 \cdot 6 \cdot 100) = -101250 / 3000 = -33$
$t_4 = (169 \cdot 25^2) / (7 \cdot 8 \cdot 100) = 20625 / 2000 = 3$

The returned value is $t_0 + t_1 + t_2 + t_3 + t_4 = 100 - 312 + 162 - 33 + 3 = -80$.

Important: since the operation of division introduces an approximation, you can do it only after computing an element of the series. In other words, you have to compute the numerator and denominator of equation (3) first, then you divide the numerator by the denominator.
The suggested method guarantees the absence of overflow during computation (this is not true for other implementations).

Computer Architectures
Exam of 12/02/2025

B2

Question 2 (8 points)

Extend the previous exercise to play the F note when the user presses the button KEY1.
In details:

- add proper initializations in `main()`
- in the handler of KEY1, start timer 1 in order to generate an interrupt every k = 1592 clock cycles
- in IRQ_timer.c, define the global variable `int cosineValues[45];`
- in the handler of timer 1, implement the following pseudocode:

```
static int repeat = 0;
static int ticks = 0;
int input, output;

if (repeat < 200)
{
     input = cast_to_int(1.428 * ticks);     // see Note 1 below
     output = 500 + Maclaurin_cos(input, 3) / 2;
     cosineValues[ticks + 22] = output;      // see Note 2 below
     write output to proper bits of D/A converter register
     ticks ++;
     if (ticks > 22)
     {
          ticks = -22;
          repeat += 1;
     }
else
     write zero to proper bits of D/A converter register
```

Note 1: for a better accuracy, before casting a float value to an integer value you should add 0.5 to the float value if it is positive. Instead, if the float value is negative, you should subtract 0.5.
Note 2: The values stored in the array `cosineValues` are not used, but they must be present. They will be checked during the evaluation of the exam.

You do not need to handle consecutive presses of KEY1: pressing it a second time has no effect. Therefore, button debouncing is not required for this project.