APAI2025 - LAB01  
**PULP\_Embedded\_Programming**

short line

*Authors: Davide Nadalini, Lorenzo Lamberti, Alberto Dequino, Manuele Rusci, Francesco Conti*

*Contacts: d.nadalini@unibo.it, alberto.dequino@unibo.it*

***Links:*** [***GitHub Link (code)***](https://github.com/EEESlab/APAI25-LAB01-PULP-Embedded-Programming)

# **Summary**

1. Subject(s):
   * PULP architecture, vector sum, matrix-vector mul, profiling code execution;
2. Programming Language: C;
3. Lab duration: 3h
4. Objective: Embedded programming & profiling

you will learn the basics of embedded programming, the pulp architecture, basic operations (sum & matmul), and how to profile your code execution (MAC, cycles) !

1. Assignment:
   * 2 tasks
   * Time for delivery: 2 weeks
   * **Submission deadline: Oct 10th 2025**

# 

# **How to deliver the assignment**

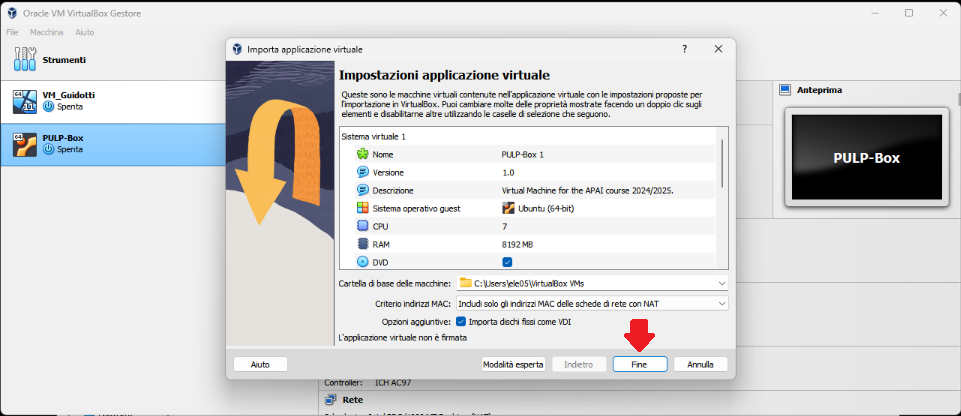
You will deliver ONLY THIS TEXT FILE, no code

* Download this file.
* Fill in the required results.
* Export to pdf format.
* Rename the file to: LAB<number\_of\_the\_lesson>\_APAI\_<your\_name>.pdf
* Use Virtuale platform to load ONLY your .pdf file

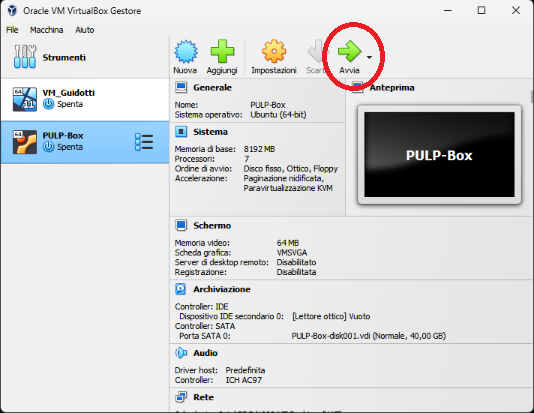
# **LAB STARTS HERE**

# **0. (ONLY LAB1) Access to the local VM**

* On the lab’s PCs, open the file explorer and go to This PC, C:/VM\_APAI
* Double click on PULP-box.ova
* VirtualBox opens, just click on “Fine”

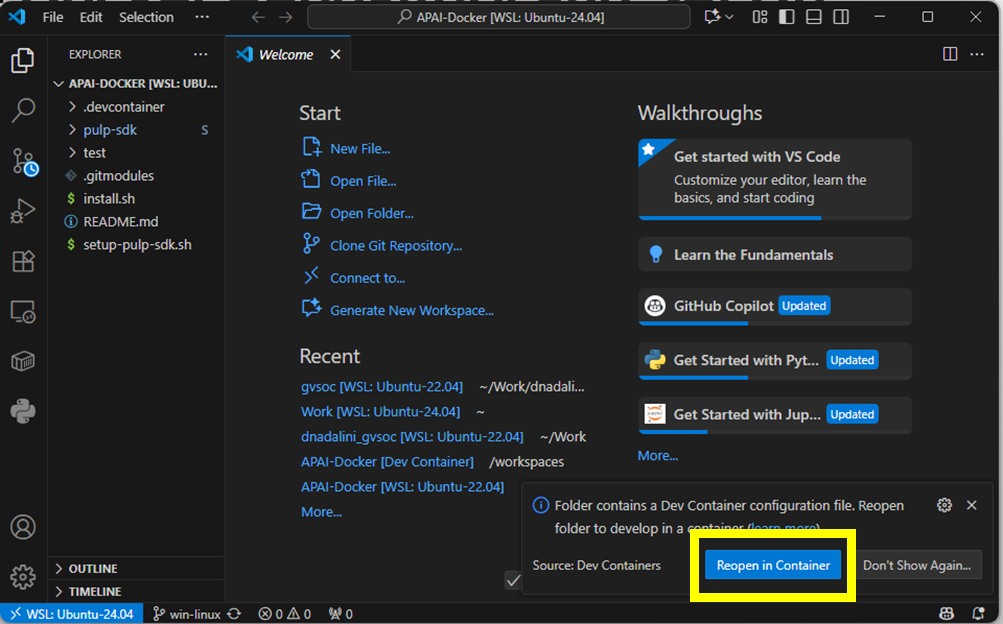


* Wait for the VM to be imported
* Open the VM with “Avvia”



* Password is ‘pulp’

# **1. (ALL) Open Docker Container with VSCode**

* Open a terminal (right click – open a new terminal)
* Go in the APAI-Docker folder: cd APAI-Docker
* Open Docker with VSCode: code .
* Select the pop-up option “Reopen in container”
* Setup PULP-SDK: source setup-pulp-sdk.sh
* Clone GitHub repository of today’s lab: git clone https://github.com/EEESlab/APAI25-LAB01-PULP-Embedded-Programming
* cd APAI25-LAB01-PULP-Embedded-Programming
* cd pulp-helloworld
* make clean all run

If the output is a “Hello World!”, then everything is correctly set up!

# **Task 1: vector sum**

**0. Setup:**

* Go to “vector\_sum/” folder
* Every time you want to run the code, **SAVE your file** and write in the terminal “**make clean all run**”

### **1.1. Define N to 50 (N= vector size) and run the code:**

|  |  |  |
| --- | --- | --- |
|  | Question | Anwer |
| 1 | What’s the result of the vector\_sum() function? |  |
| 2 | Is the checksum correct ? |  |
| 3 | Print all the elements of “array\_1”.  What’s the output? |  |

***Tips:***

* ***Question 3:*** *to print the element of the array, uncomment the function call “print\_array(array\_1, N)” inside the main()*

### **1.2. Define N to 350 (N= vector size) and run the code:**

|  |  |  |
| --- | --- | --- |
|  | Question | Anwer |
| 1 | What’s the result of the vector\_sum() function? |  |
| 2 | Is the checksum correct ? |  |
| 3 | Print all the elements of “array\_1”.  What’s the output? |  |
| 4 | Array\_1 should be filled with increasing values from 1 to N. Why isn’t the case here? |  |

***Tips:***

* ***Question 3:*** *to print the element of the array, uncomment the function call “print\_array(array\_1, N)” inside the main()*
* ***Question4:*** *what range of values a “char” data type can represent?*

### **1.3. Fix the issue: cast “array\_1” and the function’s arguments to int (or short int)**

Write down your solution

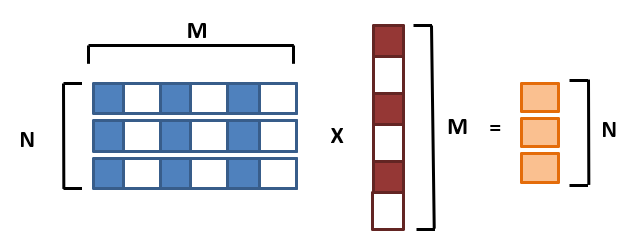
|  |  |
| --- | --- |
| **before** | **After your fix** |
| unsigned char array\_1[N]; |  |
| int init\_array(unsigned char \* A\_ar, int size) |  |
| void print\_array(unsigned char \* A\_ar, int size) |  |
| int vector\_sum(unsigned char \* A\_ar, int size) |  |

After the fix, answer again

|  |  |  |
| --- | --- | --- |
|  | Question | Anwer |
| 1 | What’s the result of the vector\_sum() function? |  |
| 2 | Is the checksum correct ? |  |
| 3 | Print all the elements of “array\_1”.  What’s the output? |  |

# 

# **Task 2: matrix-vector product & profiling**

****

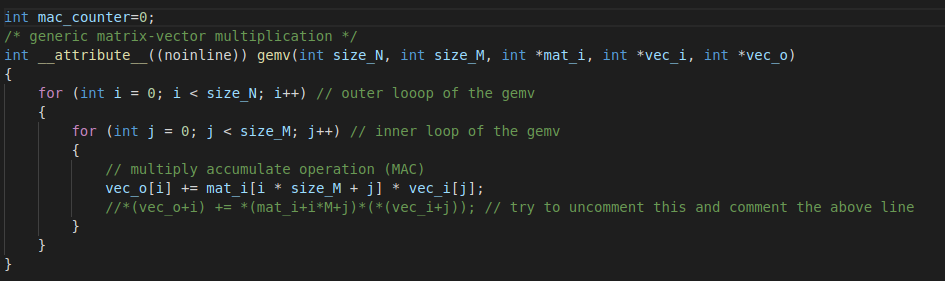
**0. Setup:**

* Go to “matrix\_vector\_product/” folder
* Every time you want to run the code, **SAVE your file** and write in the terminal “**make clean all run**”

### **2.1. Implement missing code**

GOAL: We want to count how many Multiply-ACcumulate operations we perform for a GEMV

TASK: Increment a counter every time we perform a MAC inside the GEMV inner loop. I prepared a variable called mac\_counter. This is the reference code



Put your solution below (code)

**[HERE]**

### **2.2. Implement missing code part II**

Add performance counters to profile the gemv. The gemv is calculated with this line of code in the main():

|  |
| --- |
| gemv(N, M, matrix, vector, output\_vec); |

To profile it, exploit these two functions I prepared for you. You should start the profiling right before and stop it right after

|  |
| --- |
| start\_perf\_counter()*;* stop\_perf\_counter()*;* |

### 

Put your solution below (code)

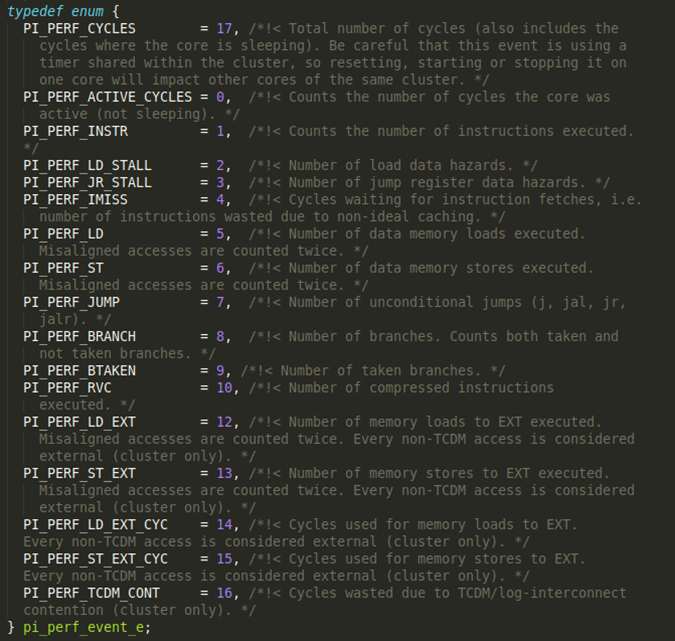
**[HERE]**

### **2.3. Implement missing code part III**

Enable the performance counters of our interest. We want to profile:

* Execution cycles (total)
* N° instructions executed

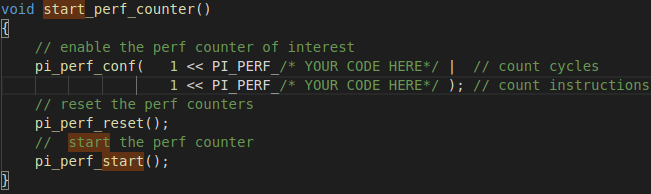
Here’s the full list of the performance counters



*Ref: /rtos/pmsis/pmsis\_api/include/pmsis/chips/default.h*

You can enable them in the start\_perf\_counter() function.

Complete the code where you find /\* YOUR CODE HERE \*/ with the right performance counters (see previous figure)



Put your solution below (code)

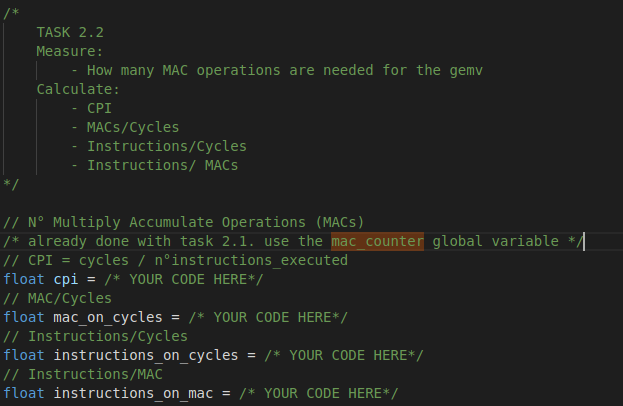
**[HERE]**

### **2.4. Implement missing code part IV**

GOAL: we now are able to measure Cycles & instructions (thanks to performance counters), and MAC operations (your implementation in the code)

TASK: calculate and print the following metrics in the code (fill in where you read /\*YOUR CODE HERE\*/)

* CPI = cycles/instructions
* MAC/cycles
* Instructions/Cycles
* Instructions/MACs



Put your solution below (code)

**[HERE]**

### **2. The size of the matrix is NxM=50x50 and the vector is M=50. Run the code and fill the table:**

Note:

* Profile the matrix-vector product with different compiler optimizations: -O1, -O3, -O3 with HW Loops (default is -O1)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **-01** | **-03 -mnohwloops** | **-03** |
| Cycles |  |  |  |
| N°Instructions |  |  |  |
| MACs |  |  |  |
| CPI |  |  |  |
| Instructions/Cycles |  |  |  |
| Instructions/MACs |  |  |  |

short dash