UNIVERSITEIT.STELLENBOSCH.UNIVERSITY jou kennisvennoot
• your knowledge partner

Computer Systems 414

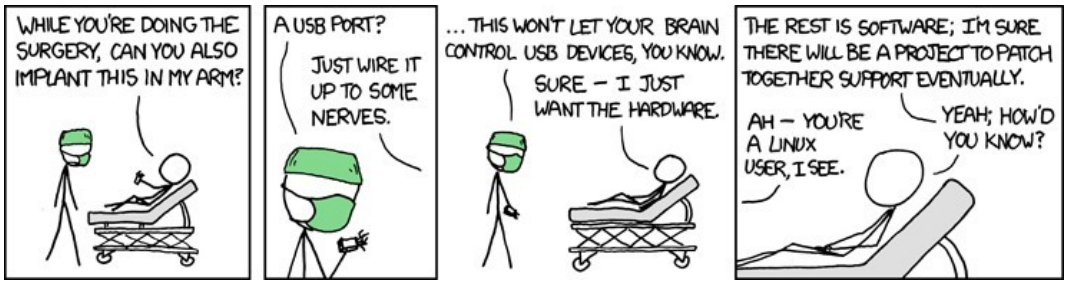
Practical 2

2021

Aim of Practical 2:

a To develop an idea of the size of the ARM ecosystem

b To gain experience with an ARM development environment



Rules of Engagement

1 Attendance as per the guidelines in the studyguide

2 Work **individually**.

3 Obtain all relevant supporting documentation on learn.sun.ac.za

4 Hand in part 2A of this assignment before 16:00 on the day of the practical (31 Mar

2021)

5 The report of part 3B is to be submitted on learn.sun.ac.za. The deadline for the reports are the afternoon (14:00) Wednesday, 7 April 2021. **No late submissions will be accepted.**

6 Do not forget to cite and give credit for any information reported which is not your property.

7 Google is your friend. Any information not given is left out on purpose. Search your solution and on the internet or relevant manuals and documentation.

Assignment 2A

Complete the following questions from the end of the relevant chapters of Wolf and submit to learn.sun.ac.za.

Chapter 7: 2, 5

Chapter 3: 1, 9, 26, 27, 29

Assignment 2B

1.1 The Keil environment is loaded on your computer - Keil uVision5

1.2 Use the Keil User Guide (uv4 and make susre that you understand the environment and that you can create applications as well as debug them. Pay special attention to Chapter 2, 3, 5, 6 and 7.) 3. Open the example project:

4.a Goto: Project -> Manage -> Pack Installer

4.b Left panel, choose "Devices", click om ARM -> ARM Cortex M3

4.c Right panel, choose "Examples", "Copy" the "DSP_Lib Sinus/Cosinus example" to your favourite local destination on the computer.

4.d In your file manager double-click on "arm_sin_cos_example.uvprojx" to open the project.

4.e "Build" and "Debug" the project. Step through the code to get a feeling for the complexity.

b.5 Now create your own project:

5.a Project -> New uVision Project

5.b Choose where you want to save your project and give it the name "sum" Click Save

5.c Choose ARM->ARM Cortex M3->ARMCM3 as "device" - Click OK

5.d Click block next to: Device -> Startup (v1.0.1) - notice the orange!

(e) Click "resolve" - This adds needed dependant parts - notice the green! (f) Click OK

b.6 Look at the framework code at the end of the instructions and the code within the example project, use it for the following steps.

b.7 Add to the project by writing C-code with the following functionality: in the main() function set two numbers (ints) and sends them to a function mulac() that multiplies the two numbers and adds it to a total (type static int) that accumulates over time. Then do the following investigation on your code:

7.a Make sure the "debug device" is set to "Simulator" -> right-click on Target1 -> Options -> Debug tab -> select "Use Simulator".

7.b Test if the code works correctly by stepping (debug mode) through the code on c-statement level and take note of how the variable values change (Call Stack + Locals window).

7.c Determine how the "static" variable is implemented. At what address in memory is it stored?

7.d Now use the "Performance Analyzer" and determine how much time (%) each functions used of the total execution time. Repeat the analysis after you changed the code to call the mulac() function 100 times from main().

7.e How many assembler instructions do the assembled version of your code contain?

b.8 Use the _asm function of the C-compiler to write an assembler routine and add it to your code with the name sum(). The routine should receive two parameters (ints), add them together and return the answer. Adapt your main() to also use the new routine.

b.9 List ALL your final code in your report.

Framework code (a.k.a. fill in the dots)

```
#define

... sum(..., ...)
{
int ans;
__asm{
... ans, ..., ...;
}
return ans;
}



... mulac(...,...)
{

static int ... = ...;

... = (... * ...) + ...;

}



int main(void)
{

int ..., ...;

... = ...;

... = ...;
```

```
22
23    mulac(..., ...);
24
25    ...
26
27    while(...);
28
29    }
30
```