Laboratory 2 tasks

Alexander Lundqvist

**For compiling and installing on chipKit**

make clean

make

make install TTYDEV=/dev/tty.usbserial-A503WFGA

**Task 1**

**Task 2**

*What does it mean when a function does not return a value? How do you state that in a*

*program? How can then the function (or more precisely, the procedure) perform anything*

*useful?*

*How did you implement the side effect that is needed to make print\_number behave*

*correctly?*

A program that doesn’t return a value (void) means that that function does not return a value to the caller explicitly.

In assembly this would translate as not storing any value in the $v-registers.

In C we define this by declaring the function type void.

Void function(parameter){do something}

A function doesn’t necessarily need to return something to be useful. For example, we could have a function swap, that takes in two addresses and then perform value swaps with pointers. This operation performs its task in memory and it certainly does SOMETHING, but it doesn’t return any value.

To implement the print\_number function, we used a global count variable. This is most of the times considered bad practice, but for the sake of keeping the function somewhat independent, we used it.

Text

Description automatically generated

**Task 3**

*How did you represent the marking of 'prime' and 'not a prime' in the memory array?*

*Which are the main steps in the algorithm? How have you implemented these steps?*

*What is the largest prime number that you can print within 2 seconds of computation? What*

*is the largest number you can print within 10 seconds? Is it the same for print\_prime.c,*

*sieves.c, and sieves-heap.c? Why or why not?*

To represent the numbers, we use an

|  |  |  |  |
| --- | --- | --- | --- |
|  | **print\_prime.c** | **sieves.c** | **sieves-heap.c** |
| **2 seconds** |  |  |  |
| **10 seconds** |  |  |  |

*This was performed on a macbook pro retina 13’’ (2013)*

**Task 4**

*Explain how you get the pointer addresses to the two char arrays (text1 and text2) and*

*the counter variable (count) in function work().*

*What does it mean to increment a pointer? What is the difference between incrementing the*

*pointer that points to the ASCII text string, and incrementing the pointer that points to the*

*integer array? In what way is the assembler code and the C code different?*

*What is the difference between incrementing a pointer and incrementing a variable that a*

*pointer points to? Explain how your code is incrementing the count variable.*

*Explain a statement in your code where you are dereferencing a pointer. What does this*

*mean? Explain by comparing with the corresponding assembler code.*

*Is your computer using big-endian or little-endian? How did you come to your conclusion?*

*Is there any benefit of using either of the two alternatives?*

**Task 5**

*Before the oral exam, you should prepare the answers to the following questions. You will need to*

*be able to answer these questions to pass the assignment.*

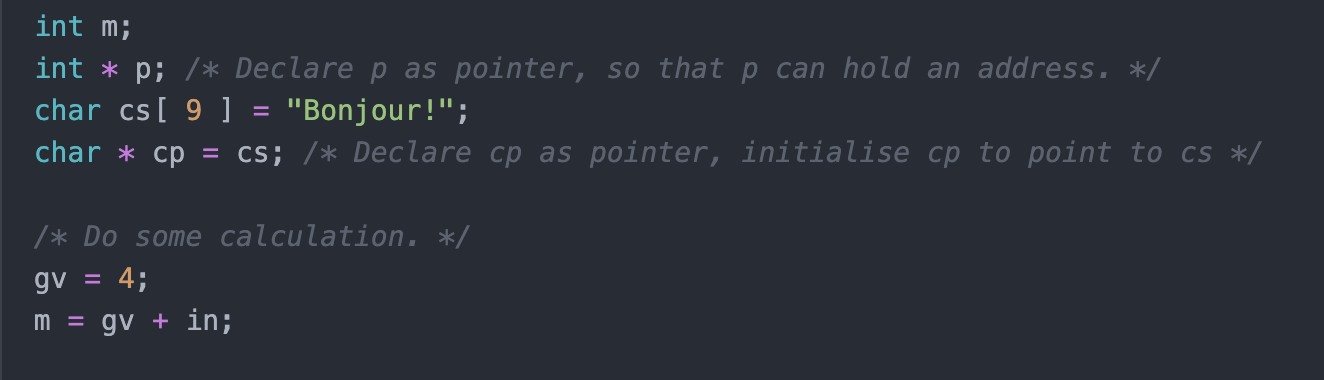
*1. Consider AM18, AM19, and AF1. Explain why gv ends up with the incremented value, but*

*m does not.*

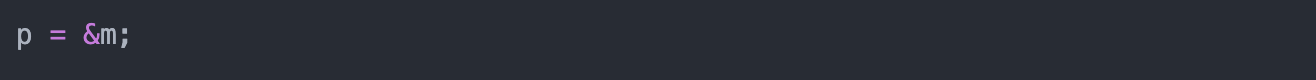
AM18 (m)

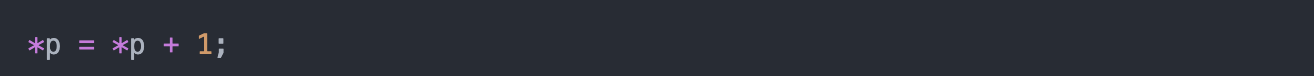
AM19 (gv)

AF1 (param)

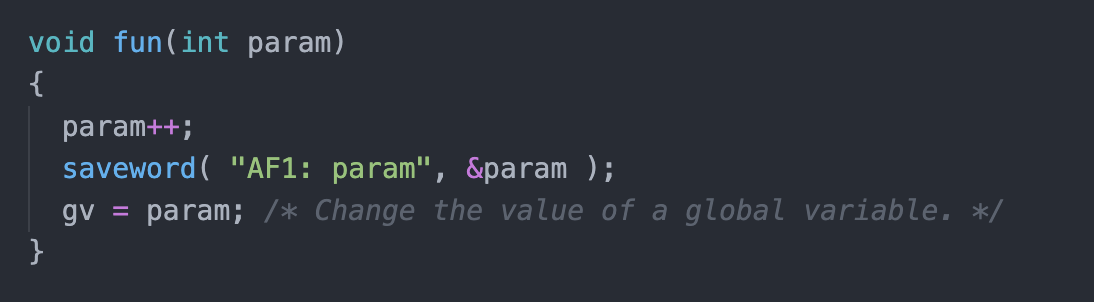


We first initialize some variables. Gv gets the value 4, and m gets gv + in (Which is a global variable initialized to 3.) -> m has the value 4+3 =7.

We then state that the pointer p holds the same address as m. This means that we can manipulate the value in memory with both p and m, since they both point to the same address.



We then dereference p and increment the value that p points to, which in turn means that when we dereference m, we will get that updated value. P&m will refer to the value 8.

We then call the function fun with m as argument.

What this does is it takes the param argument (m) and increases it with 1. Then saves this value at the address &param. We then declare that gv will have this new value (9).

Since we don’t modify m in memory, it will not get the updated value.

*2. Pointer cp is a character pointer that points to a sequence of bytes. What is the size of the*

*cp pointer itself?*

4 Bytes. A pointer refers to an address and the addresses have the format 0xnnnnnnnn, which is 32 bits.

*3. Explain how a C string is laid out in memory. Why does the character string that cp points*

*to have to be 9 bytes?*

To signal that a string has ended, we need a NUL byte, hence we have 9 instead of 8.

*4. Which addresses have fun and main? Which sections are they located in? What kind of*

*memory are they stored in? What is the meaning of the data that these symbols points to?*

*Before the examination, you should also try to answer the following. When the lab-assistant*

*performs the examination, he/she can also clarify anything that you did not understand with the*

*following questions:*

*5. Which addresses are variables in and gv located at? Which memory sections according to*

*the PIC32 memory map? Why?*

*6. Variables p and m are not global variables. Where are they allocated? Which memory*

*section is used for these variables? Why are the address numbers for p and m much larger*

*than for in and gv?*

*7. At print statement AM5, what is the address of pointer p, what is the value of pointer p, and*

*what value is pointer p pointing to?*

*8. At print statement AM7, what is the address of pointer p, what is the value of pointer p, and*

*what value is pointer p pointing to?*

*9. Consider AM14 to AM17. Is the PIC32 processor using big-endian or little-endian? Why?*

**Task 6**

*Surprise assignment!*