ANSI C

TEST

ESPINOSA BERNAL GIOVANNI
FUENTES FLORES LORENA
MARTÍNEZ OLVERA JUDITH
UGALDE ROMERO DULCE CAROLINA
VITE GONZÁLEZ CYNTHIA

MENTOR: CONTINENTAL SUPPORT TEAM

INTRODUCTION

The C language is characterized by a large number of properties that make it ideal for scientific and management uses.
C language is structured and its excellent standard library of functions, make C one of the best programming languages.
During our stay in Continetal we learned the use of ANSI C Pointers and Secure Coding.
We reviewed about the memory allocation including Static, Automatic and Dynamic Memory Allocation.
All this information was useful to solve the following test.

METHODOLOGY

The development of this test was divided in two main parts.
The first part consisted in solving the test before taking the classes of ANSI C and without any type of aid.
The second part consisted on solving the test and relate the results with the information we got in the class.

REFERENCES

Boelen, M. (15 de 05 de 2019). Linux Audit. Obtenido de The 101 of ELF files on Linux: Understanding and Analysis: https://linux-audit.com/elf-binaries-on-linux-understanding-and-analysis/
Díez, I. (11 de 06 de 2017). S3 Lab. Obtenido de ¿Cómo funciona un linker?: http://s3lab.deusto.es/como-funciona-linker-1/
Mukundan, G. (27 de 12 de 2015). Embedded Related. Obtenido de Analyzing the Linker Map file with a little help from the ELF and the DWARF: https://www.embeddedrelated.com/showarticle/900.php
UNKNOWN.(08 de 05 de 2012). cppreference. Obtenido de #pragma: https://es.cppreference.com/w/cpp/preprocessor/pragma UNKNOWN.(11 de 10 de 2013). IBM. Obtenido de Exploring the DWARF debug format information: https://developer.ibm.com/articles/au-dwarf-debug-format/
UNKNOWN. (2019). Computer Hope. Obtenido de Linker: https://www.computerhope.com/legal.htm
UNKNOWN. (2019). Tutorials Point. Obtenido de C - Preprocessors: https://www.tutorialspoint.com/cprogramming/c_preprocessors.htm
UNKNOWN. (2019). What Is. Obtenido de Compiler: https://whatis.techtarget.com/definition/compiler
UNKNOWN. (s.f.). Lenguaje ANSI C. Obtenido de Guía de referencia: https://www.mhe.es/universidad/informatica/8448145127/archivos/apendice_general_2.pdf

ANSI C
TEST

ESPINOSA BERNAL GIOVANNI
FUENTES FLORES LORENA
MARTÍNEZ OLVERA JUDITH
UGALDE ROMERO DULCE CAROLINA
VITE GONZÁLEZ CYNTHIA

MENTOR: CONTINENTAL SUPPORT TEAM

RESULTS

1. **How much memory needs to be allocated upon definition of the following variables/pointers on a 32-bit microcontroller with native support for single precision floating-point:**

a) `UINT32 ARRAY[] = {1, 0x10, 0xaabbccdd, 0x1000};` — 16 bits

b) `float float_var;` — 4 bits

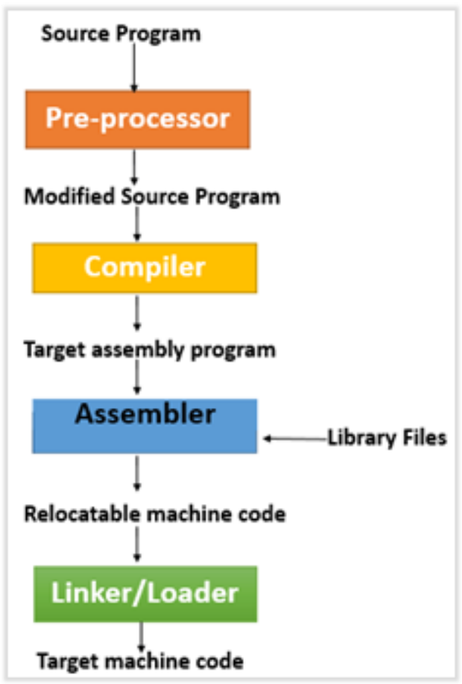
c) `UINT16 *var_prt;` — 4 bits

2. **Define the following terms:**

a) C-Preprocessor

The C-Preprocessor (**CPP**) is a separate step in the compilation process. It is just a text substitution tool and it instructs the compiler to do required pre-processing before the actual compilation.
All preprocessor commands begin with a hash symbol (#).
Examples:

- #define
- #include
- #if



b) C-Compiler

A C-complier is just compiler for programs written in C.

Compiler: A compiler is a special program that processes statements written in a particular programming language and turns them into machine language or "code" that a computer's processor uses.
When executing (running), the compiler first parses (or analyzes) all of the language statements syntactically one after the other and then builds the output code.

c) Linker

A linker is a computer program that takes one or more object files generated by a compiler and combines them into one, executable program.

3. **Define the following terms:**

a) ELF/DWARF file

DWARF (Debugging With Attributed Record Formats) is a debugging file format used by many compilers and debuggers to support source-level debugging. It is the format of debugging information within an object file. The DWARF description of a program is a tree structure where each node can have children or siblings. The nodes might represent types, variables, or functions.
ELF (Executable and Linkable Format) and defines the structure for binaries, libraries, and core files. The formal specification allows the operating system to interpreter its underlying machine instructions correctly. ELF files are typically the output of a compiler or linker and are a binary format.

b) Object file

An object file is a file containing object code, meaning relocatable format machine code that is usually not directly executable. There are various formats for object files, and the same object code can be packaged in different object files. An object file may also work like a shared library.

c) MAP file

When you're writing firmware, there always comes a time when you need to check the resources consumed, perhaps because you're running out of RAM or Flash or you want to optimize something. The **map file** generated by your linker is a useful tool to aid in the resource analysis.

4. **What is the Preprocessor directive #pragma used for?**

The **#pragma preprocessor command** allows the programmer to deliver a command to the compiler that he can execute to perform certain tasks. Since the #pragm capabilities of each compiler are specific and defined by its implementation, the use of #pragma tags changes from one compiler to another. A common example would be to track the execution of a program or modify certain options in the compilation.

ANSI C

TEST

ESPINOSA BERNAL GIOVANNI
FUENTES FLORES LORENA
MARTÍNEZ OLVERA JUDITH
UGALDE ROMERO DULCE CAROLINA
VITE GONZÁLEZ CYNTHIA

MENTOR: CONTINENTAL SUPPORT TEAM

5. Which features are valid for extern declarations?

a) To preserve the value of the object between function calls.

b) To restrict the scope of a global variable or function to the compilation unit that defines it.

c) To specify that the storage of the declared object is defined in separate modules.

d) To define global variables.

e) To declare global variables.

6. Which features are valid for static declarations?

a) Has value zero before a program starts running if a program does not explicitly initialize it.

b) They are global but not fixed memory is allocated in them.

c) Fixed memory is allocated for them.

d) Are created in the stack but they never lose their values.

7. In runtime, where are the local variables created? What is their duration?

It is created in the STACK and has automatic storage duration by default (meaning it exists only while the containing block is executing).

8. When the storage class const is used… Where the variable is allocated?

a) Stack

b) Heap

c) Internal RAM

d) Internal FLASH

NOTE: It depends on computer and architecture.

9. When the storage class stack is used… Where the variable is allocated?

a) Stack

b) Heap

c) Internal RAM

d) Internal FLASH

10. Mach the left column with the correct type description in right column.

a) `int (*fn) (void);` b) `int a[10];` c) `int a;` d) `int **a;` e) `int *a;`	(**d**) A pointer to a pointer to an integer. (**b**) An array of 10 integers. (**a**) A function pointer that receives void and returns an int. (**c**) An integer. (**e**) A pointer to an integer.

11. Define the size in bytes of "my_data" type:

```
typedef union
{
  T_U16 low_part;
  T_U32 high_part;
  union_type;
}
typedef struct
{
  T_U8 speed:6;
  T_U8 light_sensor:5;
  union_type Union_var;
} my_data;
my_data type;
```

R = 8 bytes

12. What is the output of the following code for both cases, little-endian and big-endian architecture? Suppose num is allocated at address 0x40000100.

```
unsigned int num = 0x12345678;
char* pc = (char*)& num;
for(int i=0; i<4; i++)
{
  print("%p: %02x \n", pc, (unsigned char)*pc++)
}
```

Little endian

0x40000100 -> 78
0x40000101 -> 56
0x40000102 -> 34
0x40000103 -> 12

Big endian

0x40000100 -> 12
0x40000101 -> 34
0x40000102 -> 56
0x40000103 -> 78