Cπ

Edward Garcia (ewg2115), Naveen Revanna (nr2443)
Niket Kandya (nk2531), Sean Yeh (smy2112)

Introduction

C is the lingua franca of the computer world. It is a general purpose programming language and often used in systems level programming. In this project we will be implementing Cπ (Cpi) which is a subset of the C language. It will be designed to compile to ARM V6 assembly with the target platform being the Raspberry Pi (RPi) . Cπ will use the GNU assembler(as/gas), linker(ld) and linaro cross toolchain (gcc-linaro-arm-linux-gnueabihf-raspbian) for assembly to binary code generation on the RPi.

Key Features

Cπ will be an easy language to learn for those familiar with ANSI C. The generated ARM V6 assembly will be completely unoptimized.

Keywords

Cπ will contain the following keywords

`int`	`char`	`void`	`return`	`else`
`while`	`goto`	`if`	`scan`	`print`
`struct`				

Primitive Data Types

```
int:       32 bit integers
char:      8 bit character
int*:      32 bit pointer to an integer value
char*:     32 bit pointer to a character value
void*:     32 bit pointer to a castable value
```

Aggregate Data Types

These will be defined/declared as in C collecting primitive datatypes.

```
array
structure
```

Operators

Operator precedence will follow standard orders of operation and will mimic ANSI C.

```
,     For arrays, structure definition, separate expressions
[]    Array indexing
*     Unary * for pointer dereferencing
.     For accessing structure members through a structure variable
->    For accessing structure members through a structure pointer
&     Returns address of a datatype

==    Returns an int (1 if equality holds, 0 otherwise)
!=    Returns an int (0 if equality holds, 1 otherwise)
^     Bitwise XOR operator
~     Bitwise NOT operator
&     Bitwise AND of 2 numbers
|     Bitwise OR of 2 numbers
!     Logical NOT operator
&&    Logical AND operator
||    Logical OR operator
=     Assignment operator
>     greater than operator
<     less than operator
>>    Shift right operator
<<    Shift left operator
+     Addition Operator
-     Subtraction Operator
*     Multiplication Operator
/     Division Operator
%     Modulo Operator
```

Functions

Function scope will be limited to the section enclosed in braces ({ and }). A main function will exist where code execution will start. Cπ will not support passing command line arguments to a program. Function arguments will be passed by value. Function arguments and return values will support passing pointers and functions can be recursive. To communicate to an external console, print() and scan() function will be built into the language.

Control Flow

Conditional statements such as `if`, `else`, and `else if` will be implemented according to ANSI C guidelines. Labels and `goto` statements will be supported as well as `while` loops.

Variable scoping

Global datatypes will be supported if declared outside of the scope of all functions. Normal C scoping rules apply in functions and blocks.

Comments

We will be supporting both single line and block comments. Block comments cannot be nested inside of each other.

```
//    Comment
/*    Beginning of Block Comment
*/    End of Block Comment
```

Syntax Examples

HelloWorld.cpi

```
int main() {
        print("Hello World");
        return 0;
}
```

BinarySearch.cpi

```
int binary_search(int array[], int start, int end, int element) {

    if (start > end)
        return -1;
    else {
        int mid = ((start + end)/2);
        int temp = array[mid];
        if (temp == element) {
            return mid;
        } else if (temp > element) {
            return binary_search(array, start, mid - 1, element);
        } else {
            return binary_search(array, mid + 1, end, element);
        }
    }
}

int bin_search(int array[], int size, int element) {
    return binary_search(array, 0, size - 1, element);
}

int main() {
    int arr[] = {1, 2, 4, 8, 16, 32, 64, 128, 256, 512};
    int size = 10;
    int target = scan();
    int result = bin_search()
    print(“Binary search of “, target, “in arr returns”, bin_search(arr,size,target));
    return 0;
}
```