C to MSP430 Assembler Program

Written by: Alperen Başkan

May 29, 2020

1 All the Parts of The Program

Here the program can be seen and execute part by part (even all functions separately) .

[ ]:
```
# All functions control some special characters and prints the respective␣
 ↪assembly code to the *.asm file

# These lists are defined for memory the character that we want to keep
list_for_loop = []
else_list = []
print("Please enter the name of your C program with extension (YourProgram.c):")
```

[ ]:
```
''' Here a file name is requested as a string from the user, remember that the *.
 ↪c file must be
in the same directory with this *.py file
'''
my_c_program = input()
# First, we must open the C file that entered by the user
c_file = open(my_c_program, "r")
# After having C file we need to open an empty Assembly file
asm_file = open("assembly-converted.asm","w")
```

[ ]:
```
'''function_detector function gives the name of defined all int functions,and␣
 ↪everytime
the name of the function is kept in the memory'''
def function_detector(text):
    a = text.find("int")
    b = text.find("(")
    if a != -1 and b != -1 and text.find("main") == -1:
        text = text.replace("int", "").replace("\n", "").replace(" ", "")
        list1 = list(text)
        index1 = list1.index("(")
        del list1[index1:]
        new_String = "".join(map(str, list1))
        asm_file.write(new_String)
        asm_file.write(": \n\t")
```

```
        asm_file.write("SUB.W   #4, R1\n\tMOV.W   R12, 2(R1)\n\tMOV.W   R13,␣
↪@R1\n")
    if text.find("main") != -1:
        asm_file.write("main:\n\tSUB.W   #6, R1\n")
```

[ ]:
```
# sum function is looking for if any "+" character exists.
def sum(text):
    a = text.find("+")
    if a != -1 and text.find("if") == -1 and text.find("for") == -1 and text.
↪find("else") == -1:
        if text.find("return"):
            asm_file.write("\tMOV.W   2(R1), R12\n\tADD.W   @R1, R12\n\tADD.W  ␣
↪#4, R1\n\tRET\n")
```

[ ]:
```
#substract function is just like sum function , is looking for "-" character
def substract(text):
    a = text.find("-")
    if a != -1 and text.find("if") == -1 and text.find("for") == -1 and text.
↪find("else") == -1:
        if text.find("return"):
            asm_file.write("\tMOV.W   2(R1), R12\n\tSUB.W   @R1, R12\n\tADD.W  ␣
↪#4, R1\n\tRET\n")
```

[ ]:
```
#multiply function is looking for "-" character
def multiply(text):
    a = text.find("*")
    if a != -1 and text.find("if") == -1 and text.find("for") == -1 and text.
↪find("else") == -1:
        if text.find("return"):
            asm_file.write("\tMOV.W   @(R1), R13\n\tCALL   ␣
↪#__mspabi_mpyi\n\tADD.W   #4, R1\n\tRET\n")
```

[ ]:
```
# find_forloop function search the string "for" and if for exists , splits the␣
↪string by ";" character
def find_forloop(text):
    if text.find("for") != -1 and text.find("(") != -1 and text.find(")") != -1 :
        text = text.replace("for","").replace("\t","").replace("\n","").
↪replace(" ","").replace("{","").replace("(","").replace(")","")
        text = text.split(";")
        list_for_loop = text
```

[ ]:
```
# Here find_ifcommand function looks whether there is "if" string and function␣
↪is not called if there is "else" character
def find_ifcommand(text):
    if text.find("if") != -1 and text.find("(") != -1 and text.find(")") != -1␣
↪and text.find("else") == -1:
```

```
        asm_file.write("\n\tMOV.W   4(R1),R12\n\tSUB.W   2(R1), R12")
        text = text.replace("if", "").replace(" ", "").replace("\n", "").
 ↪replace("{", "").replace(" ","").replace("\t","").replace("(","").
 ↪replace(")","")
        list1 = list(text)
        if text.find(";") == -1:
            if text.find("==") != -1:
                index_for_number = text.index("==")
                number_in_if = text[index_for_number+2:]
                asm_file.write("\n\tCMP.W   #")
                asm_file.write(number_in_if)
                asm_file.write(", R12 { JNE .L10\n")
```

[ ]:
```
# Looks if there is any "else" string basically
def else_command(text):
    if text.find("else") != -1 and text.find("if") == -1:
        text = text.replace("else", "").replace(" ", "").replace("\n", "").
 ↪replace("{", "").replace(" ", "").replace("\t", "")
        else_list.append("else")
```

[ ]:
```
# assignments_func looks the assigned values and prints the respective *.asm code
def assignments_func(text):
    if text.find("=") != -1 and text.find("int") != -1 and text.find(";") != -1 :
        text = text.replace("int", "").replace("\n", "").replace(";", "").
 ↪replace("\t", "")
        new_list =text.split(",")
        list2 = []
        for a in new_list:
            if a.find("=") != -1:
                index1 = a.index("=")
                list1 = list(a)
                del list1[0:index1+1]
                new_String = "".join(map(str, list1))
                list2.append(new_String)
        asm_file.write("\tMOV.W   #")
        asm_file.write(list2[0])
        asm_file.write(", 4(R1)\n\t")
        asm_file.write("MOV.W   #")
        asm_file.write(list2[1])
        asm_file.write(", 2(R1)\n")
    if text.find("int") == -1 and text.find("=") != -1 and text.find(";")!= -1␣
 ↪and text.find("^") == -1 and text.find("(") == -1:
        text = text.replace(" ", "").replace(";", "").replace("}","").
 ↪replace("\n","")
        index_num = text.index("=")
        list_new = list(text)
```

```
        del list_new[0:index_num+1]
        asm_file.write("\tMOV.W   #")
        asm_file.write(list_new[0])
        asm_file.write(", 4(R1)\n\tBR      #.L11\n")
```

[ ]:
```
# xor_function looks if there is any XOR operation , if exists prints XOR.W␣
 ↪operation
def xor_function(text):
    if text.find("^") != -1 and text.find("=") != -1 and text.find(";") != -1:
        asm_file.write("\tMOV.W   4(R1), R12\n\tXOR.W   2(R1), R12\n\tMOV.W  ␣
 ↪R12, @R1")
```

[ ]:
```
# or_function detector basically looks for the character "||"
def or_function(text):
    if text.find("||") != -1:
        temp_or =2
        temp1 = text[:text.find("||")]
        text = text.replace(text[:text.find("||")], "")
        text = text.replace("||", "")
        temp2 = text
        # TO DO
        asm_file.write("\tCMP.W   #0, -2(R4) { JNE      .L" + str(temp_or) +␣
 ↪"\n")
        asm_file.write("\tCMP.W   #0, -4(R4) { JEQ      .L" + str(temp_or + 1) +␣
 ↪"\n")
        asm_file.write(".L" + str(temp_or) + ":\n")
        asm_file.write("\tMOV.B   #1, R12\n")
        asm_file.write("\tBR  # .L" + str(temp_or + 2) + "\n")
        asm_file.write(".L" + str(temp_or + 1) + ":\n")
        asm_file.write("\tMOV.B   #0, R12\n")
        asm_file.write(".L" + str(temp_or + 2) + ":\n")
        asm_file.write("\tADD.W   #4, R1\n\tPOPM.W  #1, r4\n\tRET\n")
        temp_or += 5
```

[ ]:
```
#pretty same like the or function , finds if there exists any "&&" character
def and_function(text):
    if text.find("&&") != -1:
        temp_and = 1

        temp1 = text[:text.find("&&")]
        text = text.replace(text[:text.find("&&")], "")
        text = text.replace("&&", "")
        temp2 = text
        asm_file.write("\tCMP.W   #0, -2(R4) { JEQ      .L" + str(temp_and) +␣
 ↪"\n")
        asm_file.write("\tCMP.W   #0, -4(R4) { JEQ      .L" + str(temp_and) +␣
 ↪"\n")
```

```
        asm_file.write("\tMOV.B   #1, R12\n")
        asm_file.write("\tBR  # .L" + str(temp_and + 1) + "\n")
        asm_file.write(".L" + str(temp_and) + ":\n")
        asm_file.write("\tMOV.B   #0, R12\n")
        asm_file.write(".L" + str(temp_and + 1) + ":\n")
        asm_file.write("\tADD.W   #4, R1\n\tPOPM.W  #1, r4\n\tRET\n")
        temp_and += 1
```

[ ]:
```
'''
In the main function firstly , we read the C file line by line and for every␣
 ↪line we call the functions defined above.
The program controls every line with the respective functions and if there is␣
 ↪any character we looked for program writes
it to the *asm file .
'''
def main():
    for x in c_file:
        if x != "{\n":
            function_detector(x)
            sum(x)
            substract(x)
            multiply(x)
            find_ifcommand(x)
            find_forloop(x)
            else_command(x)
            assignments_func(x)
            xor_function(x)
            or_function(x)
            and_function(x)
```

[ ]:
```
'''
After calling main function , we need  to close C file first because we don't␣
 ↪need C file anymore
Since it is hard to view *.asm file , the code in the *.asm file is copied to a␣
 ↪*.txt file
Finally *asm file is closed to view our new file well.
'''
main()
c_file.close()
asm_file.close()
asm_file = open("assembly-converted.asm", "r")
x = asm_file.read()
asm_file.close()
asm2_file = open("assembly-converted.txt", "w")
asm2_file.write(x)
asm2_file.close()
```

```
print("Please check the file that you put C file to see your converted␣
 ↪(assembly-converted.asm) file\nAlso You can find the file assembly-converted.
 ↪txt to see the text in the assembly program")
print("\nYou can first try with the existing file (program.c) ")
k = input("Press Enter to Exit")
```