EED 3018 MICROPROCESSOR SYSTEMS HOMEWORK

Release 1.0

Furkan Fatih Durat

May 29, 2020

CONTENTS

Course Name: EED3018 Microprocessor Systems

Advisor: Asst.Prof. ÖZGÜR TAMER

Date: May 29, 2020

CHAPTER ONE

PURPOSE OF THE CODE

This code convert the given basic C-code to assembly-code. We can convert the summation, subtraction and multiplication operation with the two variables. Also we can use the functions calls and functions at the C-code. This code determine the and, or, and xor operations also calculate and convert these statements to assembly-code. After that we can find the assignments which are in given the C-code and it converts these assignments to assembly-code. However we can make only one assignment at each row in given C-code.

CHAPTER TWO

FUNCTIONS

"micro" is the name of the .py file

`micro.`**`and_detector`**`(`*row*`)`

"" This function finds the and operation in the given c-code. It works only all operation elements are assigned values. For example;

d = a && b (All variables a, b and d are assigned)

It finds the registers of given variables from the recorded list. It writes the necessary assembly code to creating file.

Parameters

- **`variable`** (*`String`*) – holds the variable name to which the result of and operation is assigned.
- **`variable1`** (*`String`*) – Holds the first variable name of and operation.
- **`variable2`** (*`String`*) – Holds the second variable name of and operation.

`micro.`**`assignments_detector`**`(`*row*`)`

"" This Function find the assignments which are given in the C-code and it assigns a register to each variable. Also it writes the necessary assembly code parts to creating file.Also it records the variable names and registers which are belong to these variables. .For example;

Assignment = ['a','b','c','d'. . .] (Variables)

Registers = ['&R1','2(R1)','(4R1)','6(R1)'. . .] (Registers belongs to these registers)

Parameters

- **`variable`** (*`String`*) – Holds the variable name.
- **`value`** (*`int`*) – Holds the value which assigned to variable.

`micro.`**`function_calling_detector`**`(`*row*`)`

"" This function determines the function calls at the main. Also it determines the which variables are sending and their registers. This works only when two variables are sending to function. Also it writes the necessary assembly code parts to creating file.

Parameters

- **`function_name`** (*`String`*) – Holds the calling function name.
- **`variable1`** (*`String`*) – Holds the first variable name in the function calling.
- **`variable2`** (*`String`*) – Holds the second variable name in the function calling.

`micro.`**`functions_detector`**`(`*row*`)`

“” It finds the functions in the given c-code. It writes the function name and necessary assembly code parts.

Parameters **`function_name`** (*`String`*) – Holds the function names.

`micro.`**`main`**`()`

“” We control the code from here. We send the codes to functions line by line and functions make their jobs.

Parameters **`row`** (*`String`*) – holds a row of the given C-code.

`micro.`**`multiplication_detector`**`(`*row*`)`

“” This function detects the multiplication operations. It finds the all variables at the multiplication operation and it calculates the result of multiplication for two constant variables. It writes the necessary assembly code part to creating file. It does it for two constant term. For example;

a = 2*3

Parameters

- **`variable`** (*`String`*) – holds the variable name to which the subtraction’s result is assigned.
- **`value1`** (*`int`*) – holds the value of first variable of multiplication.
- **`value2`** (*`int`*) – holds the value of second variable of multiplication.

`micro.`**`or_detector`**`(`*row*`)`

“” This function finds the or operation in the given c-code. It works only all operation elements are assigned values. For example;

d = a || b (All variables a, b and d are assigned)

It finds the registers of given variables from the recorded list. It writes the necessary assembly code to creating file.

Parameters

- **`variable`** (*`String`*) – holds the variable name to which the result of or operation is assigned.
- **`variable1`** (*`String`*) – holds the variable name to which the result of or operation is assigned.
- **`variable2`** (*`String`*) – holds the variable name to which the result of or operation is assigned.

`micro.`**`subtraction_detector`**`(`*row*`)`

This function detects the subtraction operations. It finds the all variables in the subtraction operation and it finds the result of subtraction for two constant variable. It writes the necessary assembly code parts to creating file. Also it determines the kind of variables like constant or assigned. For example;

a = b-c (subtraction of two assigned variable)

a = b-2 (subtraction of assigned and constant variables)

a = 3-2 (subtraction of two constant variables)

Parameters

- **`variable`** (*`String`*) – holds the variable name to which the subtraction’s result is assigned.
- **`variable1`** (*`String`*) – Holds the first variable name of subtraction.
- **`variable2`** (*`String`*) – Holds the second variable name of subtraction.

`micro.`**`summation_detector`**`(`*row*`)`

This function detects the summation operations. It finds the all variables in the summation operation and it finds the result of summation for two constant variable. It writes the necessary assembly code parts to creating file. Also it determines the kind of variables like constant or assigned. For example;

a = b+c (summation of two assigned variable)

a = 2+b (summation of assigned and constant variable)

a = 2+3 (summation of two constant variables)

Parameters

- **`variable`** (*`String`*) – holds the variable name to which the summation's result is assigned.
- **`variable1`** (*`String`*) – Holds the first variable name of summation.
- **`variable2`** (*`String`*) – Holds the second variable name of summation.

`micro.`**`xor_detector`**`(`*row*`)`

“” This function finds the and operation in the given c-code. It works only all operation elements are assigned values. For example;

d = a ^ b (All variables a, b and d are assigned)

It finds the registers of given variables from the recorded list. It writes the necessary assembly code to creating file.

Parameters

- **`variable`** (*`String`*) – holds the variable name to which the result of xor operation is assigned.
- **`variable1`** (*`String`*) – holds the variable name to which the result of xor operation is assigned.
- **`variable2`** (*`String`*) – holds the variable name to which the result of xor operation is assigned.

CHAPTER THREE

BUGS AND FUTURES

1. Summation, multipication and subtraction operations work only at two variable condition.
2. We can make only one asssigment at each row.
3. We can use and,or,xor statement with only two variable (a||b||c is not possible for this code).
4. We can make multiplication with only two constant terms (a=3*2 possible but a=b*c is not possible).
5. This code connot determine the if,else and for statements.
6. This code can develops to use more than two variable at mathematical operations, to use assigned variable at multplication, to use more than two variable at and,xor and or operations and we can detect if,else and for statements.

PYTHON MODULE INDEX

INDEX