Team 3
CS19B001 ADITYA SHARMA CS19B003 AMIT KESARI CS19B013 G Vaibhav

Project Report: UwU Compiler

INTRODUCTION

Our primary goal is to create a bespoke language based on the numerous programming constructs we've learned from other programming languages. The language is known as 'UwU.' To do this, we needed to create a lexer and parser that could parse the language according to the established grammar of our choice. All of the fundamental features, including assignment, comparison, conditionals, arrays, multiple expressions, looping, functions, macros and conditional statements, are supported by our language. MIPS is the target assembly code we want to use. The executable generated by Makefile is used to carry out the entire compilation and parsing process. Following are the many steps of compilation that we have incorporated into our software.

Stage	Details
Pre-Processor	Includes all content of header files and macros implementation is done.
Lexical Analysis	Tokenizes the contents of the input file (according to the language requirements).
Syntax Analysis	Makes a Parse Tree out of the tokens returned by the lexer. If an error occurs in either of the preceding two phases, the compiler does not proceed to semantic analysis.
Semantic Analysis	Creates an Abstract Syntax Tree and populates the Symbol Table for the appropriate Parse Tree. If an error is discovered during Semantic Analysis, the compiler does not generate proper code.
IR Code generation.	Construct IR code using the semantic tree's nodes stored in a synthesized attribute.
IR to MIPS:	Converts into MIPS target assembly code and register allocation.

LANGUAGE AND TOOL CHOICES

Compiler for a new language named UwU is created using lex, yacc, cpp, python and shell. Code Generation is written in python and the output is assembly code according to the MIPS architecture. This is finally run using QTSpim.

MAJOR COMPONENTS OF PROJECT

Pre-Processor

Preprocessing manipulates the text of the UwU source file, as a first phase of translation that is initiated by the compiler. Common tasks accomplished by preprocessing are macro substitution and file inclusion. Input is a program written in UwU language pgm.uwu and output is output.uwupre.

Lexer

This is the lex program, which contains numerous regular expressions for all of the particular activities that a lexical analyzer must do. This file is transformed to lex.yy.c, which is then compiled to produce the executable ./lexer. Input is a pre-processed source code output.uwupre.

Parser including Grammar

The parser has been built in the form of a yacc file called lexer.y, which implements the LALR(1) Parser and contains all of the grammar rules and actions for each production. Each production rule adheres to the S - ascribed Grammar idea (SAG).

Symbol Table

The Symbol Table includes entries for all parameters that have been declared or initialized. It is implemented using a struct array, using the variable's index as input to the Data Structure.

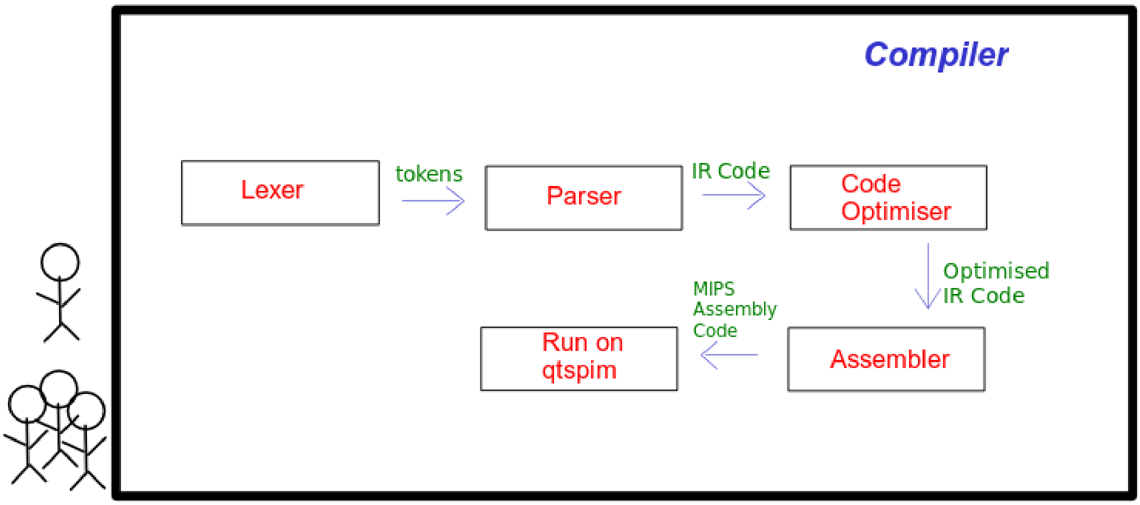
Intermediate Code Generation

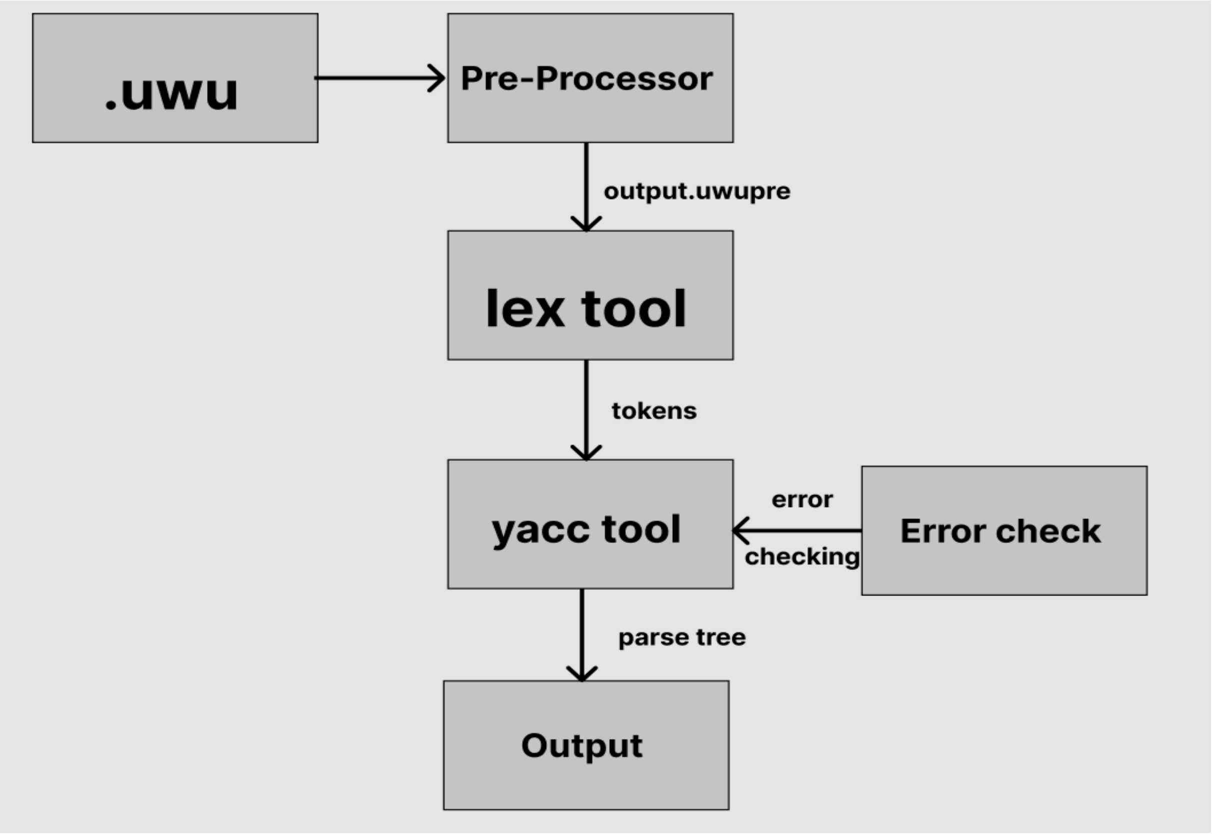
Using the semantic tree's nodes, we construct code and store it in a synthesized attribute.

MIPS Assembly Code Generation

Using the Custom IR code standard, converting the code into MIPS assembly language and running on QtSPim for simulation.

FLOW BETWEEN COMPONENTS (DIAGRAMS)





SOURCE CODE ORGANIZATION

Source Files, header files

We divided the project into modules, and each module represent a different stage of the compilation process. The working directory is made up of the following components:

- Makefile - On running it through the terminal a target parser is generated in the same working directory named ‘parser’
- Examples - The various test/sample cases using our designed grammar have been written in the files with ‘.uwu extension
- lexer.l - Lexical phase of the Compiler
- lexer.y - Parsing and Semantic analysis phase of the compiler
- Ir-to-mips.py - Takes in the IR code generated from parse tree of semantic analyzer and converting it to MIPS assembly code
- run.sh - For combining all the executables and run as a single command

Instructions

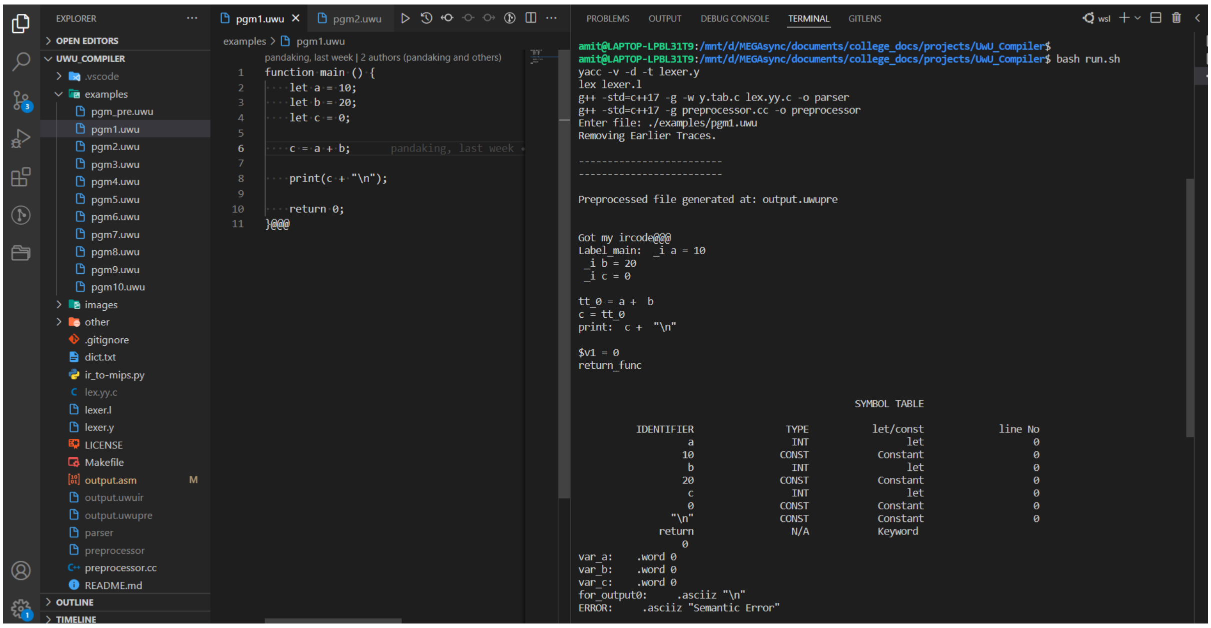
- Run the bash script ./run.sh and provide the src file name in the prompt.
- The assembly code will be written into the output.asm file of the Compiler folder. It can then be loaded into the QtSpim application from its file->Reinitialize and Load File and run the code.

OR

- Run the Makefile using make command in the terminal. On successful execution, an executable named ‘parser’ is created.
- To run this use ./parser Examples/pgm1.uwu if you have to get the corresponding assembly output in MIPS for the pgm.uwu file.
- To run the parser on any other file, the format of the terminal command should be the same replacing the filepath.
- The assembly code will be written into the output.asm file of the Compiler folder. It can then be loaded into the QtSpim application from its file->Reinitialize and Load File and run the code.

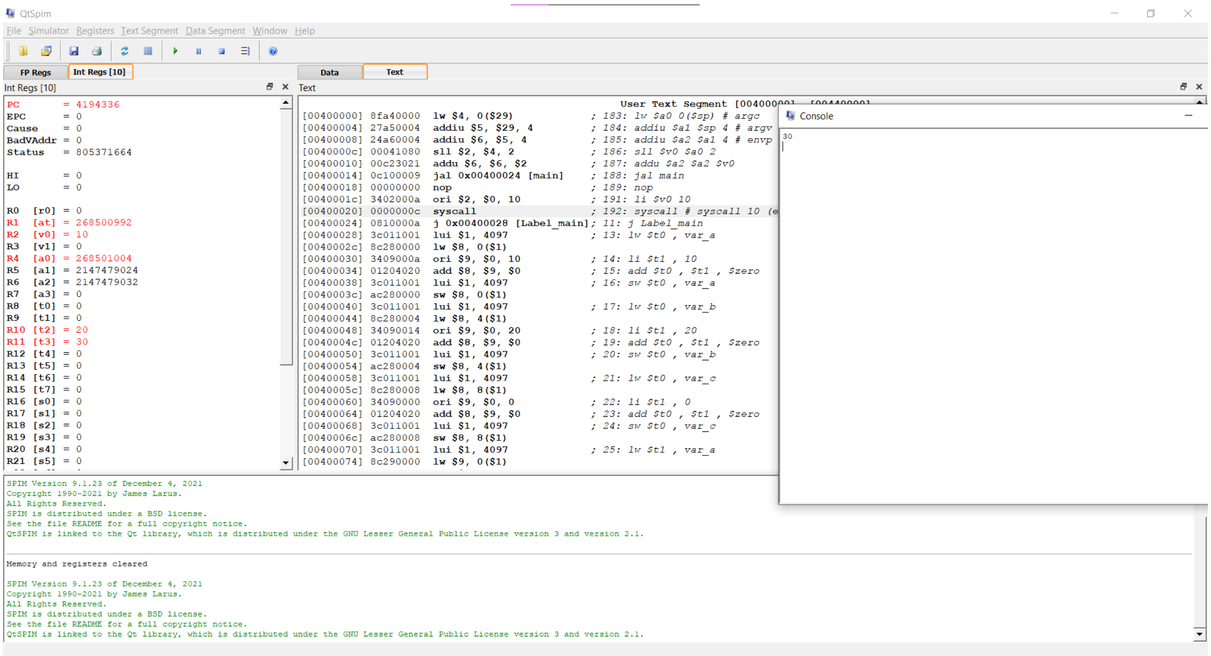
FINAL TESTING AND EVALUATION (WITH SCREENSHOTS)

- pgm1.uwu



```
function main () {
    let a = 10;
    let b = 20;
    let c = 0;

    c = a + b;

    print(c + "\n");

    return 0;
}@@@
```

```
amit@LAPTOP-LPBL31T9:/mnt/d/MEGAsync/documents/college_docs/projects/UwU_Compiler$
amit@LAPTOP-LPBL31T9:/mnt/d/MEGAsync/documents/college_docs/projects/UwU_Compiler$ bash run.sh
yacc -v -d -t lexer.y
lex lexer.l
g++ -std=c++17 -g -w y.tab.c lex.yy.c -o parser
g++ -std=c++17 -g preprocessor.cc -o preprocessor
Enter file: ./examples/pgm1.uwu
Removing Earlier Traces.

------------------------
------------------------

Preprocessed file generated at: output.uwupre


Got my ircode@@@
Label_main:  _i a = 10
 _i b = 20
 _i c = 0

tt_0 = a +  b
c = tt_0
print:  c +  "\n"

$v1 = 0
return_func


                                   SYMBOL TABLE


         IDENTIFIER               TYPE          let/const           line No
                  a                INT                let                 0
                 10              CONST           Constant                 0
                  b                INT                let                 0
                 20              CONST           Constant                 0
                  c                INT                let                 0
                  0              CONST           Constant                 0
               "\n"              CONST           Constant                 0
             return                N/A           Keyword
                 0
var_a:    .word 0
var_b:    .word 0
var_c:    .word 0
for_output0:     .asciiz "\n"
ERROR:     .asciiz "Semantic Error"
```



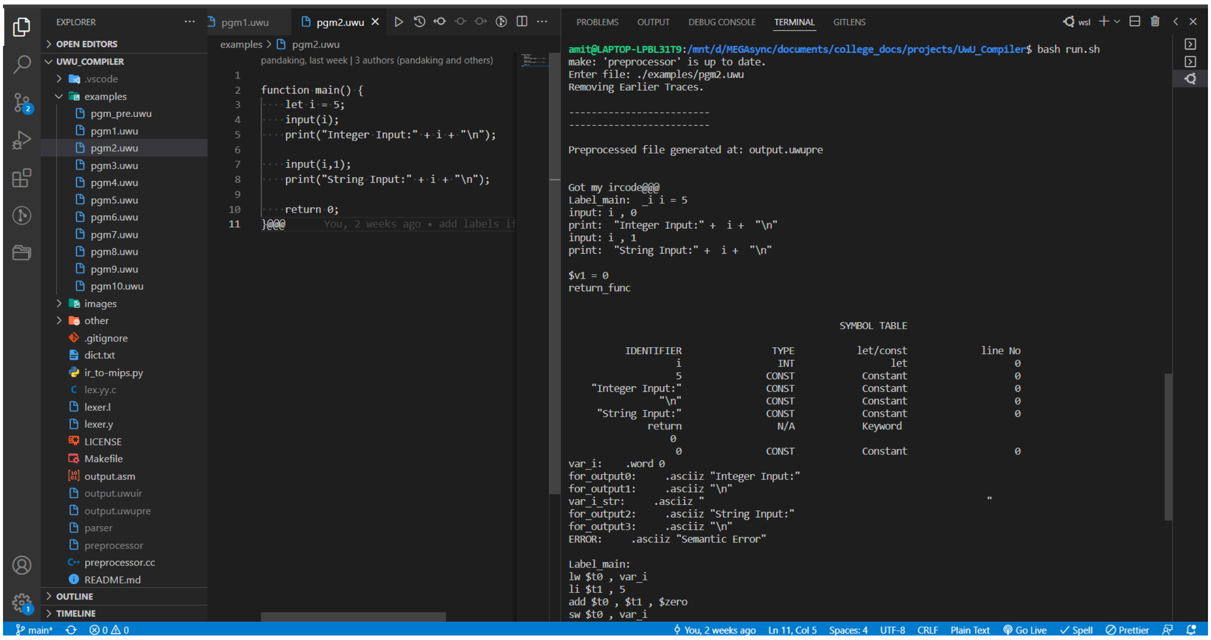
```
Int Regs [10]
PC       = 4194336
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 268501004
R5  [a1] = 2147479024
R6  [a2] = 2147479032
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 20
R11 [t3] = 30
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
```

```
Text
                                                        User Text Segment [00400000]
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4           ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4            ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2              ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2            ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]      ; 188: jal main
[00400018] 00000000  nop                        ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10             ; 191: li $v0 10
[00400020] 0000000c  syscall                    ; 192: syscall # syscall 10 (e
[00400024] 0810000a  j 0x00400028 [Label_main]; 11: j Label_main
[00400028] 3c011001  lui $1, 4097               ; 13: lw $t0 , var_a
[0040002c] 8c280000  lw $8, 0($1)
[00400030] 3409000a  ori $9, $0, 10             ; 14: li $t1 , 10
[00400034] 01204020  add $8, $9, $0             ; 15: add $t0 , $t1 , $zero
[00400038] 3c011001  lui $1, 4097               ; 16: sw $t0 , var_a
[0040003c] ac280000  sw $8, 0($1)
[00400040] 3c011001  lui $1, 4097               ; 17: lw $t0 , var_b
[00400044] 8c280004  lw $8, 4($1)
[00400048] 34090014  ori $9, $0, 20             ; 18: li $t1 , 20
[0040004c] 01204020  add $8, $9, $0             ; 19: add $t0 , $t1 , $zero
[00400050] 3c011001  lui $1, 4097               ; 20: sw $t0 , var_b
[00400054] ac280004  sw $8, 4($1)
[00400058] 3c011001  lui $1, 4097               ; 21: lw $t0 , var_c
[0040005c] 8c280008  lw $8, 8($1)
[00400060] 34090000  ori $9, $0, 0              ; 22: li $t1 , 0
[00400064] 01204020  add $8, $9, $0             ; 23: add $t0 , $t1 , $zero
[00400068] 3c011001  lui $1, 4097               ; 24: sw $t0 , var_c
[0040006c] ac280008  sw $8, 8($1)
[00400070] 3c011001  lui $1, 4097               ; 25: lw $t1 , var_a
[00400074] 8c290000  lw $9, 0($1)
```

```
Console
30
```

```
SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

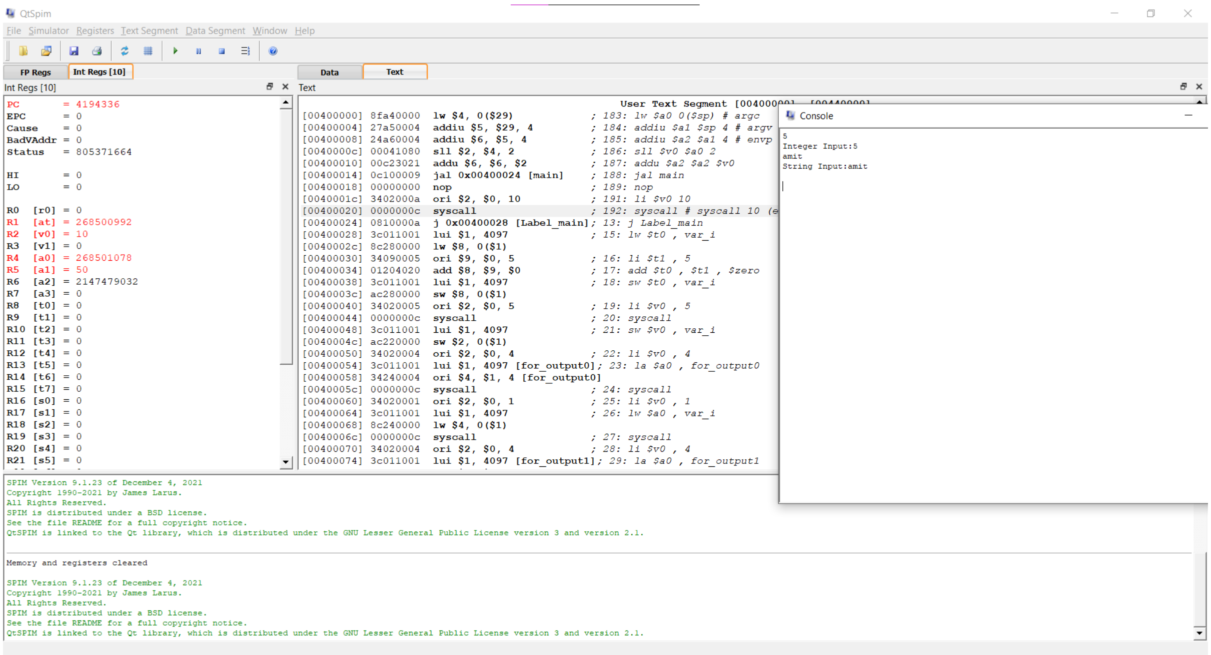
- pgm2.uwu



pandaking, last week | 3 authors (pandaking and others)

```
function main() {
    let i = 5;
    input(i);
    print("Integer Input:" + i + "\n");

    input(i,1);
    print("String Input:" + i + "\n");

    return 0;
}@@@       You, 2 weeks ago • add labels if
```

```
amit@LAPTOP-LPBL31T9:/mnt/d/MEGAsync/documents/college_docs/projects/UwU_Compiler$ bash run.sh
make: 'preprocessor' is up to date.
Enter file: ./examples/pgm2.uwu
Removing Earlier Traces.

------------------------
------------------------

Preprocessed file generated at: output.uwupre


Got my ircode@@@
Label_main:  _i i = 5
input: i , 0
print:  "Integer Input:" +  i +  "\n"
input: i , 1
print:  "String Input:" +  i +  "\n"

$v1 = 0
return_func


                                 SYMBOL TABLE

         IDENTIFIER              TYPE         let/const            line No
                  i               INT               let                  0
                  5             CONST          Constant                  0
     "Integer Input:"           CONST          Constant                  0
                "\n"            CONST          Constant                  0
      "String Input:"           CONST          Constant                  0
             return               N/A          Keyword
                 0
                  0             CONST          Constant                  0
var_i:    .word 0
for_output0:     .asciiz "Integer Input:"
for_output1:     .asciiz "\n"
var_i_str:     .asciiz "                                              "
for_output2:     .asciiz "String Input:"
for_output3:     .asciiz "\n"
ERROR:     .asciiz "Semantic Error"

Label_main:
lw $t0 , var_i
li $t1 , 5
add $t0 , $t1 , $zero
sw $t0 , var_i
```



Int Regs [10]

```
PC       = 4194336
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 268501078
R5  [a1] = 50
R6  [a2] = 2147479032
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
```

Text

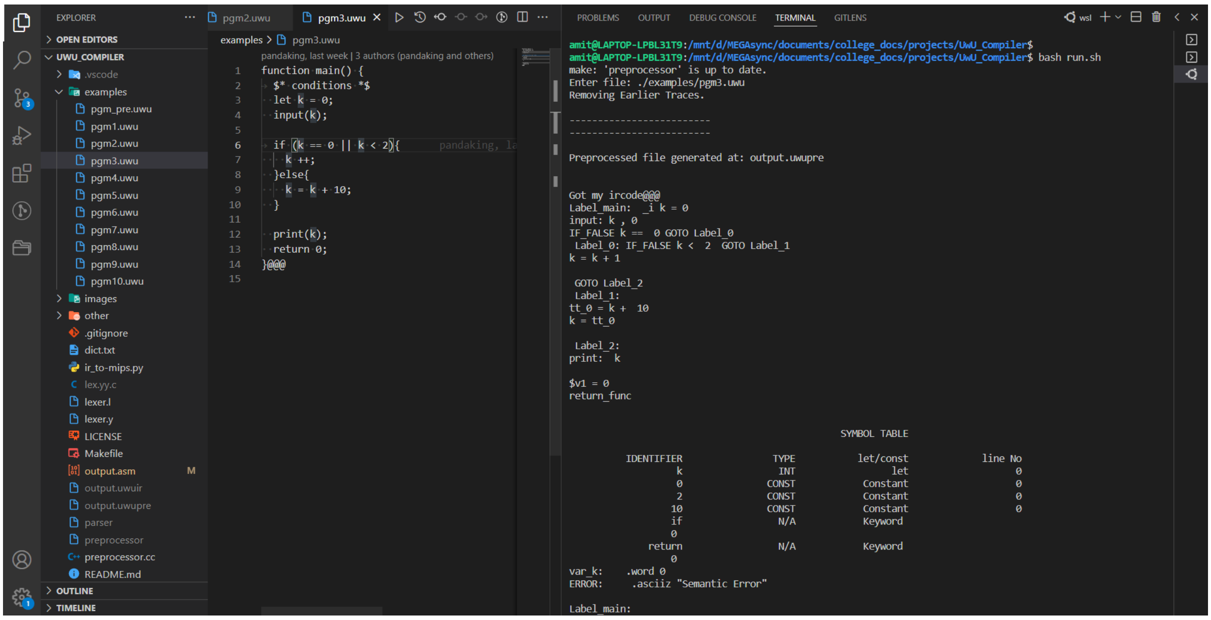
```
                                                  User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4           ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4            ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2              ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2            ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]      ; 188: jal main
[00400018] 00000000  nop                        ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10             ; 191: li $v0 10
[00400020] 0000000c  syscall                    ; 192: syscall # syscall 10 (e
[00400024] 0810000a  j 0x00400028 [Label_main]; 13: j Label_main
[00400028] 3c011001  lui $1, 4097               ; 15: lw $t0 , var_i
[0040002c] 8c280000  lw $8, 0($1)
[00400030] 34090005  ori $9, $0, 5              ; 16: li $t1 , 5
[00400034] 01204020  add $8, $9, $0             ; 17: add $t0 , $t1 , $zero
[00400038] 3c011001  lui $1, 4097               ; 18: sw $t0 , var_i
[0040003c] ac280000  sw $8, 0($1)
[00400040] 34020005  ori $2, $0, 5              ; 19: li $v0 , 5
[00400044] 0000000c  syscall                    ; 20: syscall
[00400048] 3c011001  lui $1, 4097               ; 21: sw $v0 , var_i
[0040004c] ac220000  sw $2, 0($1)
[00400050] 34020004  ori $2, $0, 4              ; 22: li $v0 , 4
[00400054] 3c011001  lui $1, 4097 [for_output0]; 23: la $a0 , for_output0
[00400058] 34240004  ori $4, $1, 4 [for_output0]
[0040005c] 0000000c  syscall                    ; 24: syscall
[00400060] 34020001  ori $2, $0, 1              ; 25: li $v0 , 1
[00400064] 3c011001  lui $1, 4097               ; 26: lw $a0 , var_i
[00400068] 8c240000  lw $4, 0($1)
[0040006c] 0000000c  syscall                    ; 27: syscall
[00400070] 34020004  ori $2, $0, 4              ; 28: li $v0 , 4
[00400074] 3c011001  lui $1, 4097 [for_output1]; 29: la $a0 , for_output1
```

Console

```
5
Integer Input:5
amit
String Input:amit
```

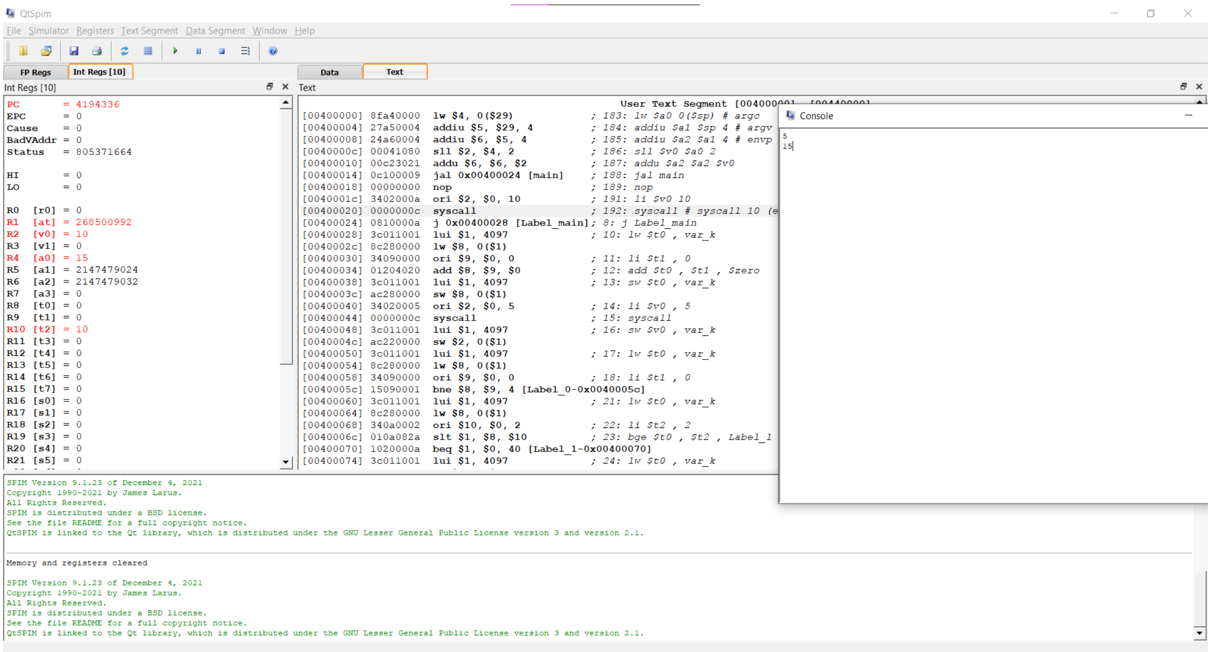
```
SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

- pgm3.uwu



```
function main() {
  $* conditions *$
  let k = 0;
  input(k);

  if (k == 0 || k < 2){
     k ++;
  }else{
     k = k + 10;
  }

  print(k);
  return 0;
}@@@
```

```
amit@LAPTOP-LPBL31T9:/mnt/d/MEGAsync/documents/college_docs/projects/UwU_Compiler$
amit@LAPTOP-LPBL31T9:/mnt/d/MEGAsync/documents/college_docs/projects/UwU_Compiler$ bash run.sh
make: 'preprocessor' is up to date.
Enter file: ./examples/pgm3.uwu
Removing Earlier Traces.

------------------------
------------------------

Preprocessed file generated at: output.uwupre


Got my ircode@@@
Label_main:  _i k = 0
input: k , 0
IF_FALSE k ==  0 GOTO Label_0
 Label_0: IF_FALSE k <  2  GOTO Label_1
k = k + 1

 GOTO Label_2
 Label_1:
tt_0 = k +  10
k = tt_0

 Label_2:
print:  k

$v1 = 0
return_func


                                   SYMBOL TABLE


         IDENTIFIER               TYPE          let/const          line No
                  k                INT                let                0
                  0              CONST           Constant                0
                  2              CONST           Constant                0
                 10              CONST           Constant                0
                 if                N/A           Keyword
                 0
             return                N/A           Keyword
                 0
var_k:    .word 0
ERROR:     .asciiz "Semantic Error"

Label_main:
```



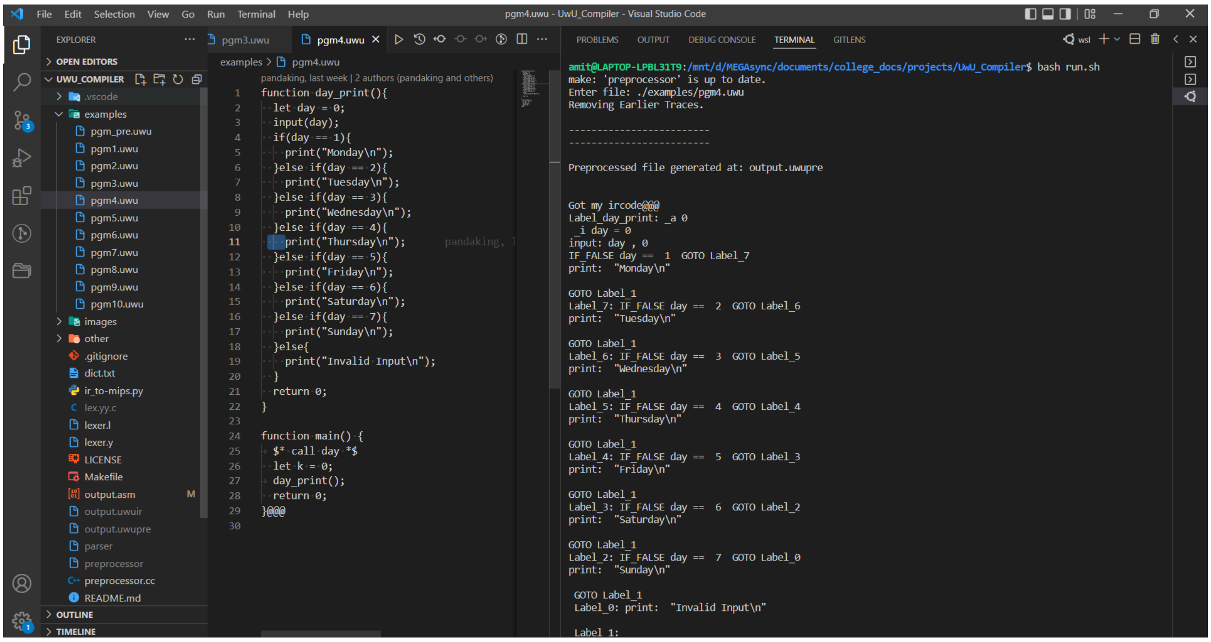
```
PC       = 4194336
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 15
R5  [a1] = 2147479024
R6  [a2] = 2147479032
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 10
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
```

```
                                                  User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)            ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4         ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4          ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2            ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2          ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]    ; 188: jal main
[00400018] 00000000  nop                      ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10           ; 191: li $v0 10
[00400020] 0000000c  syscall                  ; 192: syscall # syscall 10 (e
[00400024] 0810000a  j 0x00400028 [Label_main]; 8: j Label_main
[00400028] 3c011001  lui $1, 4097             ; 10: lw $t0 , var_k
[0040002c] 8c280000  lw $8, 0($1)
[00400030] 34090000  ori $9, $0, 0            ; 11: li $t1 , 0
[00400034] 01204020  add $8, $9, $0           ; 12: add $t0 , $t1 , $zero
[00400038] 3c011001  lui $1, 4097             ; 13: sw $t0 , var_k
[0040003c] ac280000  sw $8, 0($1)
[00400040] 34020005  ori $2, $0, 5            ; 14: li $v0 , 5
[00400044] 0000000c  syscall                  ; 15: syscall
[00400048] 3c011001  lui $1, 4097             ; 16: sw $v0 , var_k
[0040004c] ac220000  sw $2, 0($1)
[00400050] 3c011001  lui $1, 4097             ; 17: lw $t0 , var_k
[00400054] 8c280000  lw $8, 0($1)
[00400058] 34090000  ori $9, $0, 0            ; 18: li $t1 , 0
[0040005c] 15090001  bne $8, $9, 4 [Label_0-0x0040005c]
[00400060] 3c011001  lui $1, 4097             ; 21: lw $t0 , var_k
[00400064] 8c280000  lw $8, 0($1)
[00400068] 340a0002  ori $10, $0, 2           ; 22: li $t2 , 2
[0040006c] 010a082a  slt $1, $8, $10          ; 23: bge $t0 , $t2 , Label_1
[00400070] 1020000a  beq $1, $0, 40 [Label_1-0x00400070]
[00400074] 3c011001  lui $1, 4097             ; 24: lw $t0 , var_k
```

```
5
15
```

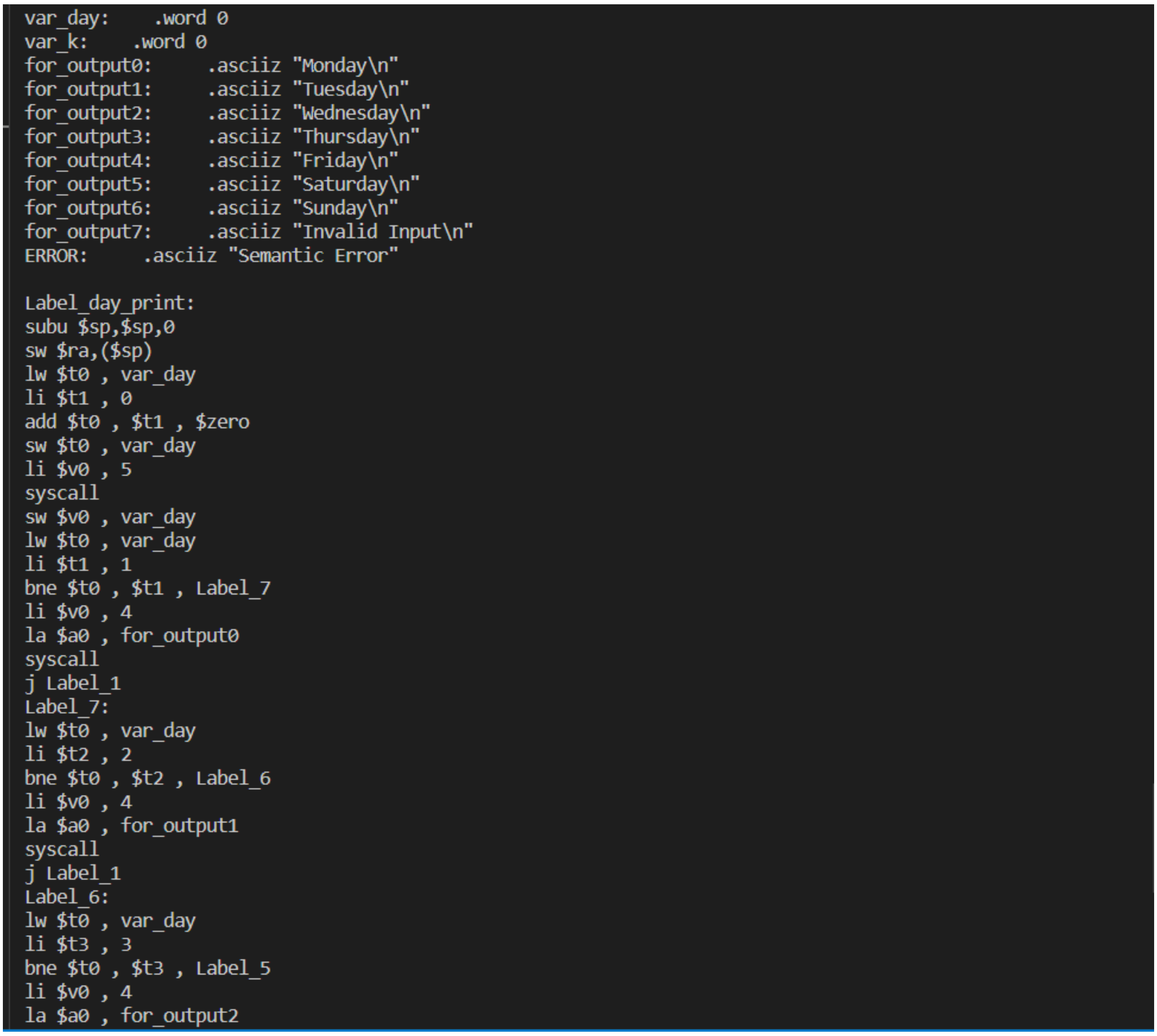
```
SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

- pgm4.uwu

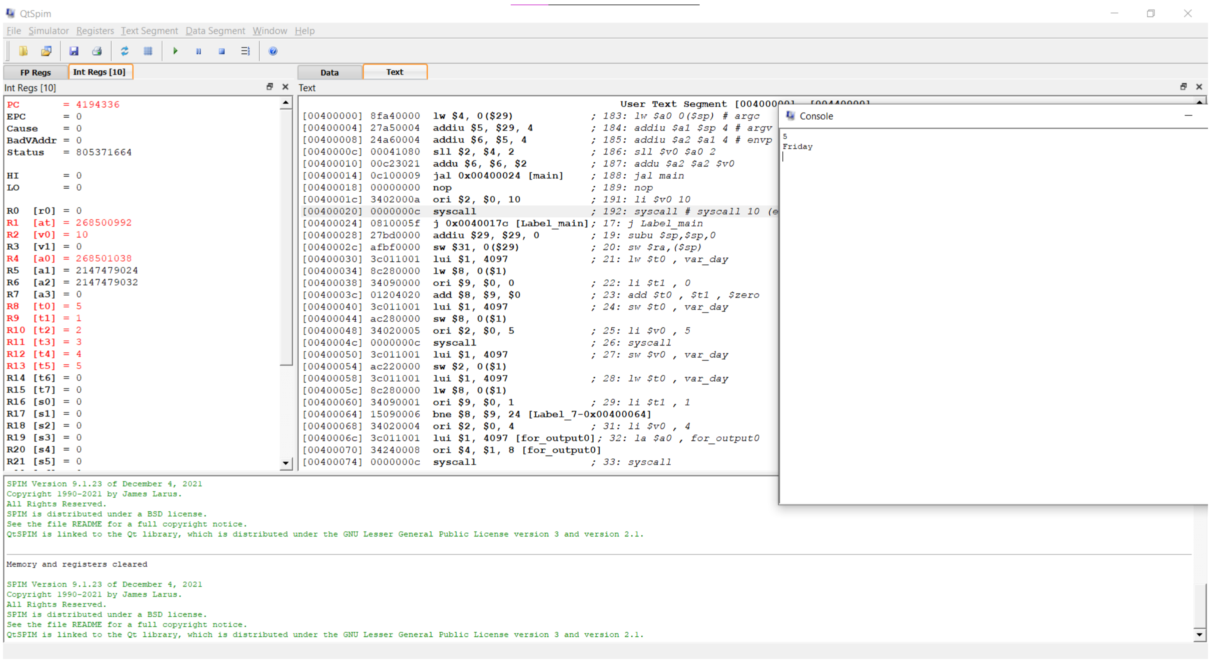


```
function day_print(){
  let day = 0;
  input(day);
  if(day == 1){
    print("Monday\n");
  }else if(day == 2){
    print("Tuesday\n");
  }else if(day == 3){
    print("Wednesday\n");
  }else if(day == 4){
    print("Thursday\n");
  }else if(day == 5){
    print("Friday\n");
  }else if(day == 6){
    print("Saturday\n");
  }else if(day == 7){
    print("Sunday\n");
  }else{
    print("Invalid Input\n");
  }
  return 0;
}

function main() {
  $* call day *$
  let k = 0;
  day_print();
  return 0;
}@@@
```

```
amit@LAPTOP-LPBL31T9:/mnt/d/MEGAsync/documents/college_docs/projects/UwU_Compiler$ bash run.sh
make: 'preprocessor' is up to date.
Enter file: ./examples/pgm4.uwu
Removing Earlier Traces.

------------------------
------------------------

Preprocessed file generated at: output.uwupre


Got my ircode@@@
Label_day_print: _a 0
 _i day = 0
input: day , 0
IF_FALSE day ==  1  GOTO Label_7
print:  "Monday\n"

GOTO Label_1
Label_7: IF_FALSE day ==  2  GOTO Label_6
print:  "Tuesday\n"

GOTO Label_1
Label_6: IF_FALSE day ==  3  GOTO Label_5
print:  "Wednesday\n"

GOTO Label_1
Label_5: IF_FALSE day ==  4  GOTO Label_4
print:  "Thursday\n"

GOTO Label_1
Label_4: IF_FALSE day ==  5  GOTO Label_3
print:  "Friday\n"

GOTO Label_1
Label_3: IF_FALSE day ==  6  GOTO Label_2
print:  "Saturday\n"

GOTO Label_1
Label_2: IF_FALSE day ==  7  GOTO Label_0
print:  "Sunday\n"

 GOTO Label_1
 Label_0: print:  "Invalid Input\n"

 Label_1:
```



```
var_day:     .word 0
var_k:     .word 0
for_output0:     .asciiz "Monday\n"
for_output1:     .asciiz "Tuesday\n"
for_output2:     .asciiz "Wednesday\n"
for_output3:     .asciiz "Thursday\n"
for_output4:     .asciiz "Friday\n"
for_output5:     .asciiz "Saturday\n"
for_output6:     .asciiz "Sunday\n"
for_output7:     .asciiz "Invalid Input\n"
ERROR:     .asciiz "Semantic Error"

Label_day_print:
subu $sp,$sp,0
sw $ra,($sp)
lw $t0 , var_day
li $t1 , 0
add $t0 , $t1 , $zero
sw $t0 , var_day
li $v0 , 5
syscall
sw $v0 , var_day
lw $t0 , var_day
li $t1 , 1
bne $t0 , $t1 , Label_7
li $v0 , 4
la $a0 , for_output0
syscall
j Label_1
Label_7:
lw $t0 , var_day
li $t2 , 2
bne $t0 , $t2 , Label_6
li $v0 , 4
la $a0 , for_output1
syscall
j Label_1
Label_6:
lw $t0 , var_day
li $t3 , 3
bne $t0 , $t3 , Label_5
li $v0 , 4
la $a0 , for_output2
```


QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [10]
Data
Text
Int Regs [10]
Text
Console

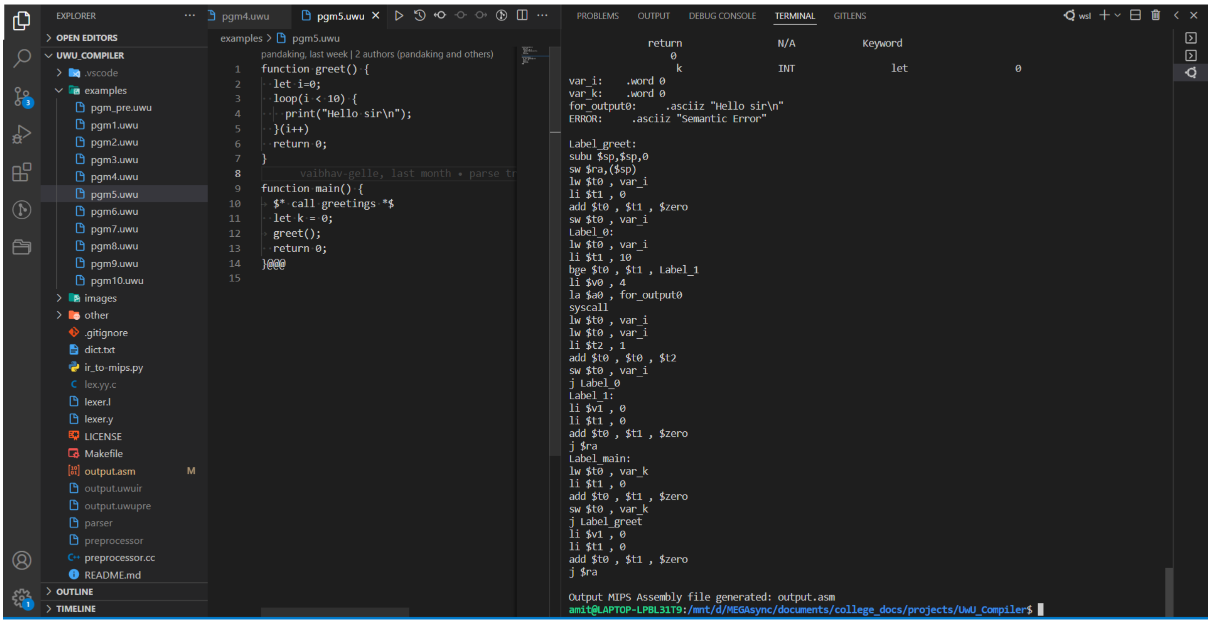
```
PC       = 4194336
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 268500992
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 268501038
R5  [a1] = 2147479024
R6  [a2] = 2147479032
R7  [a3] = 0
R8  [t0] = 5
R9  [t1] = 1
R10 [t2] = 2
R11 [t3] = 3
R12 [t4] = 4
R13 [t5] = 5
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
```

```
                                                  User Text Segment [00400000]   [00440000]
[00400000] 8fa40000  lw $4, 0($29)             ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4          ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4           ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2             ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2           ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]     ; 188: jal main
[00400018] 00000000  nop                       ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10            ; 191: li $v0 10
[00400020] 0000000c  syscall                   ; 192: syscall # syscall 10 (e
[00400024] 0810005f  j 0x0040017c [Label_main]; 17: j Label_main
[00400028] 27bd0000  addiu $29, $29, 0         ; 19: subu $sp,$sp,0
[0040002c] afbf0000  sw $31, 0($29)            ; 20: sw $ra,($sp)
[00400030] 3c011001  lui $1, 4097              ; 21: lw $t0 , var_day
[00400034] 8c280000  lw $8, 0($1)
[00400038] 34090000  ori $9, $0, 0             ; 22: li $t1 , 0
[0040003c] 01204020  add $8, $9, $0            ; 23: add $t0 , $t1 , $zero
[00400040] 3c011001  lui $1, 4097              ; 24: sw $t0 , var_day
[00400044] ac280000  sw $8, 0($1)
[00400048] 34020005  ori $2, $0, 5             ; 25: li $v0 , 5
[0040004c] 0000000c  syscall                   ; 26: syscall
[00400050] 3c011001  lui $1, 4097              ; 27: sw $v0 , var_day
[00400054] ac220000  sw $2, 0($1)
[00400058] 3c011001  lui $1, 4097              ; 28: lw $t0 , var_day
[0040005c] 8c280000  lw $8, 0($1)
[00400060] 34090001  ori $9, $0, 1             ; 29: li $t1 , 1
[00400064] 15090006  bne $8, $9, 24 [Label_7-0x00400064]
[00400068] 34020004  ori $2, $0, 4             ; 31: li $v0 , 4
[0040006c] 3c011001  lui $1, 4097 [for_output0]; 32: la $a0 , for_output0
[00400070] 34240008  ori $4, $1, 8 [for_output0]
[00400074] 0000000c  syscall                   ; 33: syscall
```

```
5
Friday
```

```
SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

- pgm5.uwu

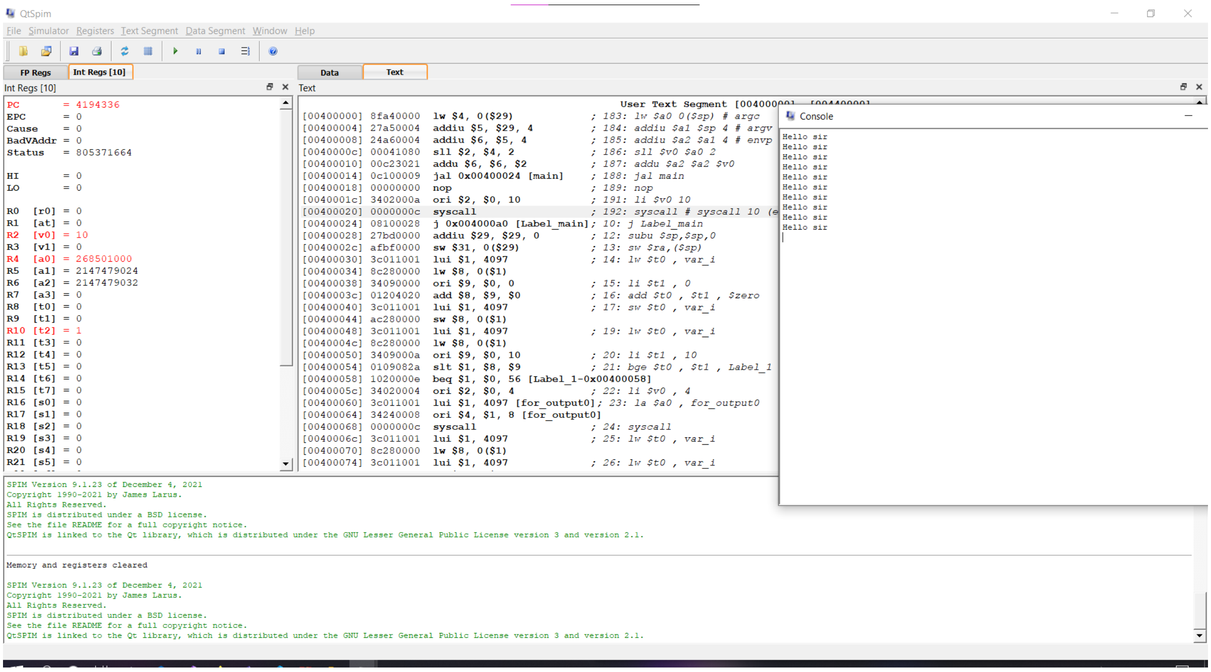


examples > pgm5.uwu

pandaking, last week | 2 authors (pandaking and others)

```
function greet() {
  let i=0;
  loop(i < 10) {
    print("Hello sir\n");
  }(i++)
  return 0;
}
        vaibhav-gelle, last month • parse tr
function main() {
  $* call greetings *$
  let k = 0;
  greet();
  return 0;
}@@@
```

```
             return              N/A          Keyword
                0
                 k               INT              let               0
var_i:    .word 0
var_k:    .word 0
for_output0:     .asciiz "Hello sir\n"
ERROR:     .asciiz "Semantic Error"

Label_greet:
subu $sp,$sp,0
sw $ra,($sp)
lw $t0 , var_i
li $t1 , 0
add $t0 , $t1 , $zero
sw $t0 , var_i
Label_0:
lw $t0 , var_i
li $t1 , 10
bge $t0 , $t1 , Label_1
li $v0 , 4
la $a0 , for_output0
syscall
lw $t0 , var_i
lw $t0 , var_i
li $t2 , 1
add $t0 , $t0 , $t2
sw $t0 , var_i
j Label_0
Label_1:
li $v1 , 0
li $t1 , 0
add $t0 , $t1 , $zero
j $ra
Label_main:
lw $t0 , var_k
li $t1 , 0
add $t0 , $t1 , $zero
sw $t0 , var_k
j Label_greet
li $v1 , 0
li $t1 , 0
add $t0 , $t1 , $zero
j $ra

Output MIPS Assembly file generated: output.asm
amit@LAPTOP-LPBL31T9:/mnt/d/MEGAsync/documents/college_docs/projects/UwU_Compiler$
```



```
PC       = 4194336
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 268501000
R5  [a1] = 2147479024
R6  [a2] = 2147479032
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 1
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
```

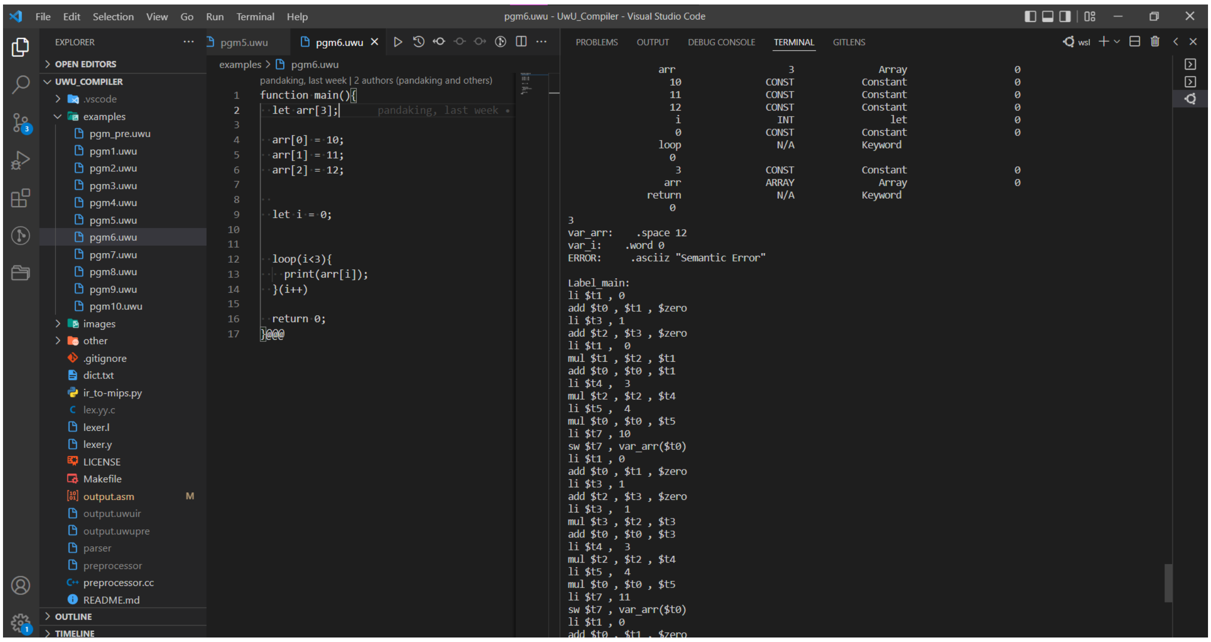
```
                                              User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)            ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4         ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4          ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2            ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2          ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]    ; 188: jal main
[00400018] 00000000  nop                      ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10           ; 191: li $v0 10
[00400020] 0000000c  syscall                  ; 192: syscall # syscall 10 (e
[00400024] 08100028  j 0x004000a0 [Label_main]; 10: j Label_main
[00400028] 27bd0000  addiu $29, $29, 0        ; 12: subu $sp,$sp,0
[0040002c] afbf0000  sw $31, 0($29)           ; 13: sw $ra,($sp)
[00400030] 3c011001  lui $1, 4097             ; 14: lw $t0 , var_i
[00400034] 8c280000  lw $8, 0($1)
[00400038] 34090000  ori $9, $0, 0            ; 15: li $t1 , 0
[0040003c] 01204020  add $8, $9, $0           ; 16: add $t0 , $t1 , $zero
[00400040] 3c011001  lui $1, 4097             ; 17: sw $t0 , var_i
[00400044] ac280000  sw $8, 0($1)
[00400048] 3c011001  lui $1, 4097             ; 19: lw $t0 , var_i
[0040004c] 8c280000  lw $8, 0($1)
[00400050] 3409000a  ori $9, $0, 10           ; 20: li $t1 , 10
[00400054] 0109082a  slt $1, $8, $9           ; 21: bge $t0 , $t1 , Label_1
[00400058] 1020000e  beq $1, $0, 56 [Label_1-0x00400058]
[0040005c] 34020004  ori $2, $0, 4            ; 22: li $v0 , 4
[00400060] 3c011001  lui $1, 4097 [for_output0]; 23: la $a0 , for_output0
[00400064] 34240008  ori $4, $1, 8 [for_output0]
[00400068] 0000000c  syscall                  ; 24: syscall
[0040006c] 3c011001  lui $1, 4097             ; 25: lw $t0 , var_i
[00400070] 8c280000  lw $8, 0($1)
[00400074] 3c011001  lui $1, 4097             ; 26: lw $t0 , var_i
```

Console

```
Hello sir
Hello sir
Hello sir
Hello sir
Hello sir
Hello sir
Hello sir
Hello sir
Hello sir
Hello sir
```

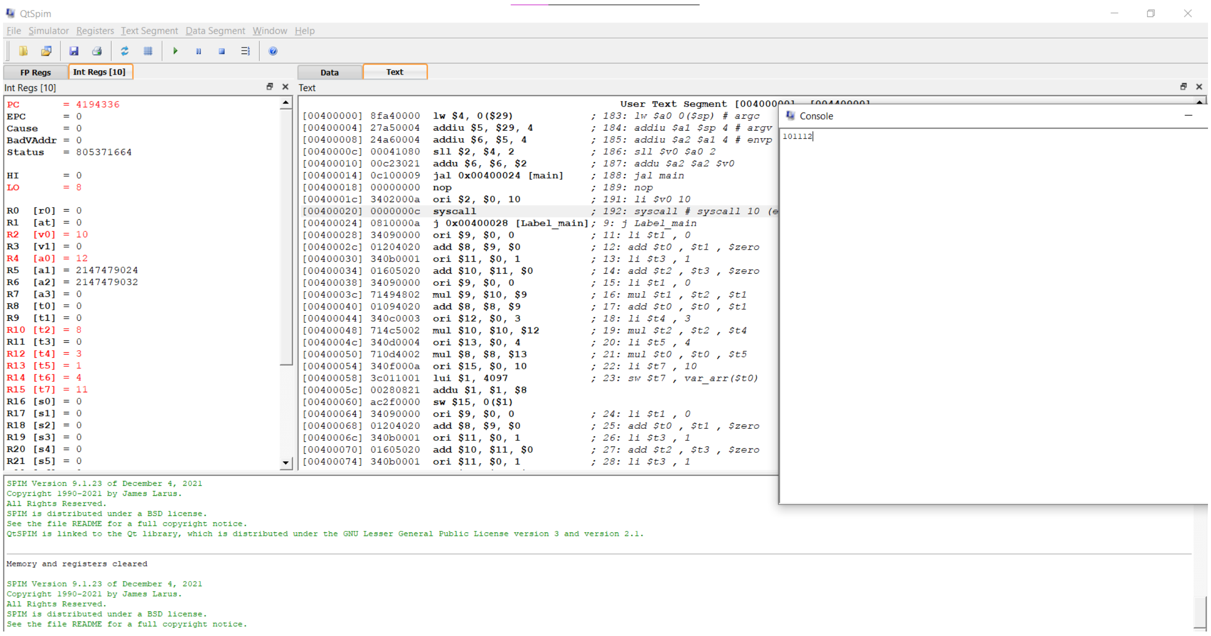
```
SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

- pgm6.uwu



```
function main(){
  let arr[3];

  arr[0] = 10;
  arr[1] = 11;
  arr[2] = 12;


  let i = 0;


  loop(i<3){
    print(arr[i]);
  }(i++)

  return 0;
}@@@
```

```
            arr                  3          Array          0
             10              CONST       Constant          0
             11              CONST       Constant          0
             12              CONST       Constant          0
              i                INT            let          0
              0              CONST       Constant          0
           loop                N/A        Keyword
             0
              3              CONST       Constant          0
            arr              ARRAY          Array          0
         return                N/A        Keyword
             0
3
var_arr:    .space 12
var_i:    .word 0
ERROR:     .asciiz "Semantic Error"

Label_main:
li $t1 , 0
add $t0 , $t1 , $zero
li $t3 , 1
add $t2 , $t3 , $zero
li $t1 ,  0
mul $t1 , $t2 , $t1
add $t0 , $t0 , $t1
li $t4 ,  3
mul $t2 , $t2 , $t4
li $t5 ,  4
mul $t0 , $t0 , $t5
li $t7 , 10
sw $t7 , var_arr($t0)
li $t1 , 0
add $t0 , $t1 , $zero
li $t3 , 1
add $t2 , $t3 , $zero
li $t3 ,  1
mul $t3 , $t2 , $t3
add $t0 , $t0 , $t3
li $t4 ,  3
mul $t2 , $t2 , $t4
li $t5 ,  4
mul $t0 , $t0 , $t5
li $t7 , 11
sw $t7 , var_arr($t0)
li $t1 , 0
add $t0 , $t1 , $zero
```


QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [10]
Data
Text
Console

Int Regs [10]

```
PC       = 4194336
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 8

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 12
R5  [a1] = 2147479024
R6  [a2] = 2147479032
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 8
R11 [t3] = 0
R12 [t4] = 3
R13 [t5] = 1
R14 [t6] = 4
R15 [t7] = 11
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
```

Text

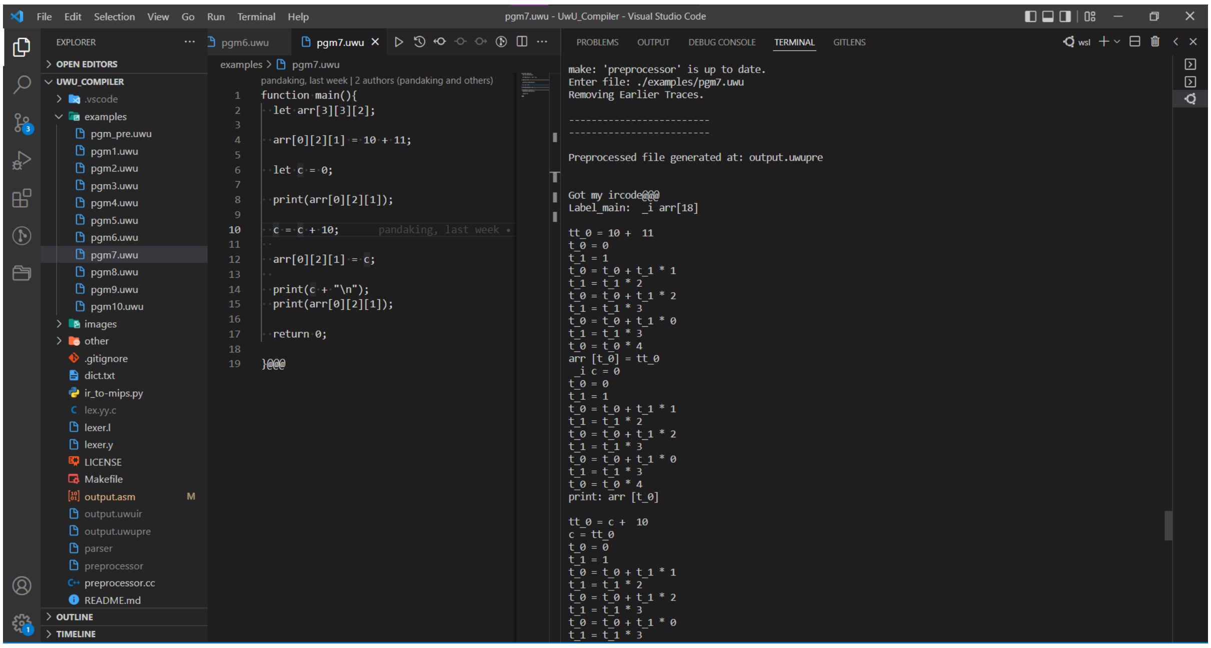
```
                                                    User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4           ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4            ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2              ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2            ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]      ; 188: jal main
[00400018] 00000000  nop                        ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10             ; 191: li $v0 10
[00400020] 0000000c  syscall                    ; 192: syscall # syscall 10 (e
[00400024] 0810000a  j 0x00400028 [Label_main]; 9: j Label_main
[00400028] 34090000  ori $9, $0, 0              ; 11: li $t1 , 0
[0040002c] 01204020  add $8, $9, $0             ; 12: add $t0 , $t1 , $zero
[00400030] 340b0001  ori $11, $0, 1             ; 13: li $t3 , 1
[00400034] 01605020  add $10, $11, $0           ; 14: add $t2 , $t3 , $zero
[00400038] 34090000  ori $9, $0, 0              ; 15: li $t1 , 0
[0040003c] 71494802  mul $9, $10, $9            ; 16: mul $t1 , $t2 , $t1
[00400040] 01094020  add $8, $8, $9             ; 17: add $t0 , $t0 , $t1
[00400044] 340c0003  ori $12, $0, 3             ; 18: li $t4 , 3
[00400048] 714c5002  mul $10, $10, $12          ; 19: mul $t2 , $t2 , $t4
[0040004c] 340d0004  ori $13, $0, 4             ; 20: li $t5 , 4
[00400050] 710d4002  mul $8, $8, $13            ; 21: mul $t0 , $t0 , $t5
[00400054] 340f000a  ori $15, $0, 10            ; 22: li $t7 , 10
[00400058] 3c011001  lui $1, 4097               ; 23: sw $t7 , var_arr($t0)
[0040005c] 00280821  addu $1, $1, $8
[00400060] ac2f0000  sw $15, 0($1)
[00400064] 34090000  ori $9, $0, 0              ; 24: li $t1 , 0
[00400068] 01204020  add $8, $9, $0             ; 25: add $t0 , $t1 , $zero
[0040006c] 340b0001  ori $11, $0, 1             ; 26: li $t3 , 1
[00400070] 01605020  add $10, $11, $0           ; 27: add $t2 , $t3 , $zero
[00400074] 340b0001  ori $11, $0, 1             ; 28: li $t3 , 1
```

Console

```
101112
```

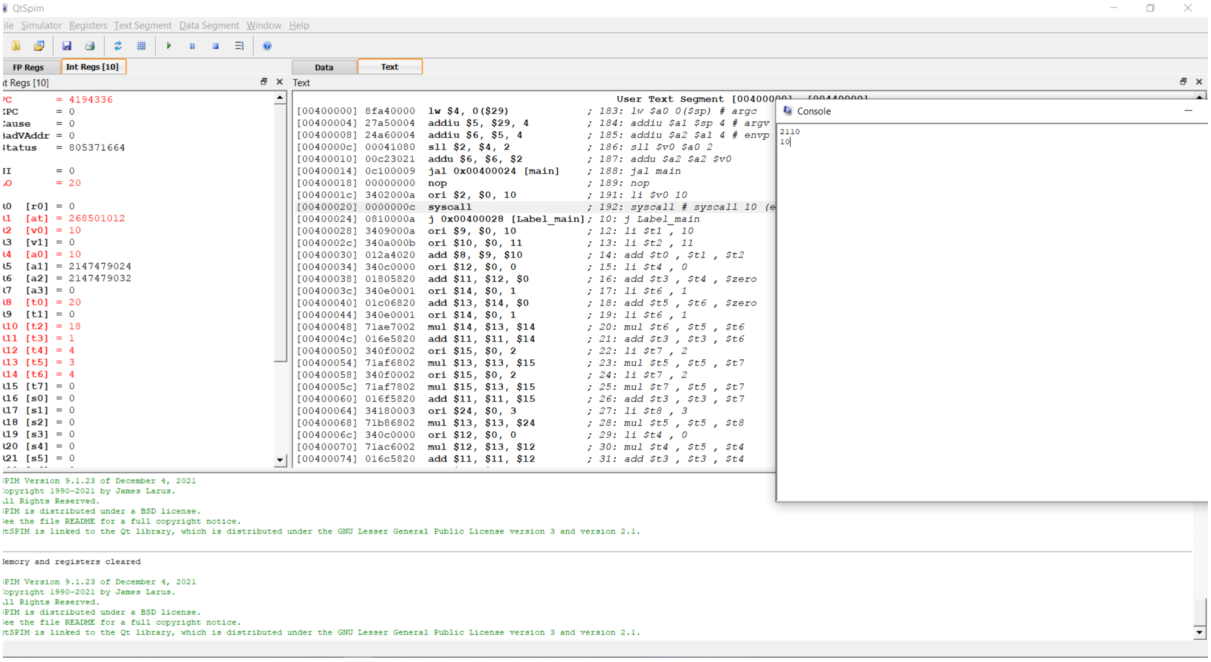
```
SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
```

- pgm7.uwu



```
function main(){
  let arr[3][3][2];

  arr[0][2][1] = 10 + 11;

  let c = 0;

  print(arr[0][2][1]);

  c = c + 10;        pandaking, last week •

  arr[0][2][1] = c;

  print(c + "\n");
  print(arr[0][2][1]);

  return 0;

}@@@
```

```
make: 'preprocessor' is up to date.
Enter file: ./examples/pgm7.uwu
Removing Earlier Traces.

------------------------
------------------------

Preprocessed file generated at: output.uwupre


Got my ircode@@@
Label_main:  _i arr[18]

tt_0 = 10 +  11
t_0 = 0
t_1 = 1
t_0 = t_0 + t_1 * 1
t_1 = t_1 * 2
t_0 = t_0 + t_1 * 2
t_1 = t_1 * 3
t_0 = t_0 + t_1 * 0
t_1 = t_1 * 3
t_0 = t_0 * 4
arr [t_0] = tt_0
 _i c = 0
t_0 = 0
t_1 = 1
t_0 = t_0 + t_1 * 1
t_1 = t_1 * 2
t_0 = t_0 + t_1 * 2
t_1 = t_1 * 3
t_0 = t_0 + t_1 * 0
t_1 = t_1 * 3
t_0 = t_0 * 4
print: arr [t_0]

tt_0 = c +  10
c = tt_0
t_0 = 0
t_1 = 1
t_0 = t_0 + t_1 * 1
t_1 = t_1 * 2
t_0 = t_0 + t_1 * 2
t_1 = t_1 * 3
t_0 = t_0 + t_1 * 0
t_1 = t_1 * 3
```



```
PC       = 4194336
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 20

R0  [r0] = 0
R1  [at] = 268501012
R2  [v0] = 10
R3  [v1] = 0
R4  [a0] = 10
R5  [a1] = 2147479024
R6  [a2] = 2147479032
R7  [a3] = 0
R8  [t0] = 20
R9  [t1] = 0
R10 [t2] = 18
R11 [t3] = 1
R12 [t4] = 4
R13 [t5] = 3
R14 [t6] = 4
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
```

```
                                              User Text Segment [00400000]  [00440000]
[00400000] 8fa40000  lw $4, 0($29)             ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4          ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4           ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2             ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2           ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]     ; 188: jal main
[00400018] 00000000  nop                       ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10            ; 191: li $v0 10
[00400020] 0000000c  syscall                   ; 192: syscall # syscall 10 (e
[00400024] 0810000a  j 0x00400028 [Label_main]; 10: j Label_main
[00400028] 3409000a  ori $9, $0, 10            ; 12: li $t1 , 10
[0040002c] 340a000b  ori $10, $0, 11           ; 13: li $t2 , 11
[00400030] 012a4020  add $8, $9, $10           ; 14: add $t0 , $t1 , $t2
[00400034] 340c0000  ori $12, $0, 0            ; 15: li $t4 , 0
[00400038] 01805820  add $11, $12, $0          ; 16: add $t3 , $t4 , $zero
[0040003c] 340e0001  ori $14, $0, 1            ; 17: li $t6 , 1
[00400040] 01c06820  add $13, $14, $0          ; 18: add $t5 , $t6 , $zero
[00400044] 340e0001  ori $14, $0, 1            ; 19: li $t6 , 1
[00400048] 71ae7002  mul $14, $13, $14         ; 20: mul $t6 , $t5 , $t6
[0040004c] 016e5820  add $11, $11, $14         ; 21: add $t3 , $t3 , $t6
[00400050] 340f0002  ori $15, $0, 2            ; 22: li $t7 , 2
[00400054] 71af6802  mul $13, $13, $15         ; 23: mul $t5 , $t5 , $t7
[00400058] 340f0002  ori $15, $0, 2            ; 24: li $t7 , 2
[0040005c] 71af7802  mul $15, $13, $15         ; 25: mul $t7 , $t5 , $t7
[00400060] 016f5820  add $11, $11, $15         ; 26: add $t3 , $t3 , $t7
[00400064] 34180003  ori $24, $0, 3            ; 27: li $t8 , 3
[00400068] 71b86802  mul $13, $13, $24         ; 28: mul $t5 , $t5 , $t8
[0040006c] 340c0000  ori $12, $0, 0            ; 29: li $t4 , 0
[00400070] 71ac6002  mul $12, $13, $12         ; 30: mul $t4 , $t5 , $t4
[00400074] 016c5820  add $11, $11, $12         ; 31: add $t3 , $t3 , $t4
```

```
2110
10
```

```
SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Memory and registers cleared

SPIM Version 9.1.23 of December 4, 2021
Copyright 1990-2021 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

LIMITATIONS OF THE COMPILER

Our code has some limitations including

- Does not contain the constructs for Classes & structs
- No Pointers
- No call by reference available

CONCLUSION

We were able to implement different components of compiler design and wrote lex and yacc files using the Bison package. We have tried to explore the actual phases of Compiler by implementing them practically. We have also modularized our project to the greatest extent feasible so that these Modules can operate independently of one another and can be improved in the future. The Code generation part has also been made isolated so that any target could be made from the generated Tree. We were able to take advantage of this feature and quickly switch our target assembly code from X86 to MIPS. Overall, the job was completed as a collaborative effort, and we utilized GitHub to share the source and manage task and role distributions.

CONTRIBUTIONS

Aditya - Language Specification Planning, Generating CFGs, Implementing Syntax analyzer, Writing report, MIPS Machine Code Generation, Testing our compiler with heavy codes, QtSpim.
Amit - Implementing Lexical analyzer, Testing Generating Syntax tree, Error detecting implementation, IR code Generation, Simulating MIPS on QtSpim.
Vaibhav - Symbol Table, Parse Tree, Checking the generated tokens, Doing register allocation, Sample programs, User and developer manual.