Lab 9 Generating MIPS CODE
Yuhao Lan

Levels:
a) Can only read and write variables and numbers 40% off
b) A program which calls no functions and does no array arithmetic at least 30% off
c) Can do a-b and "if" and "while" 20% off
d) Can do a-c and also arrays 10%
e) Can do a-d and handle function calls
f) Add strings to "write", Extra Credit 5%... You MUST do a-e do get the extra credit

I have finished a) b) c) and part of e). So I think I will get 85% of the whole scores.

TEST ONE:

```
void main (int b)
{
    read b;
    if (b > 10)
      write 1;
    else
      write 0;
}
```

the output:

```
2
3  main:
4  subu $sp,$sp,4 #substract the stack pointer for main to get space for global variables
5  addi $gp,$sp,0 #copy the space to global pointer for global variables
6  subu $sp, $sp,16 #substract the stack pointer for main
7  subu $t2,$sp,16 #get the space of function
8  sw $ra,($t2) #Store the return address
9  sw $sp,4($t2) #Store the old stack pointer
10 addu $sp,$t2,0 #copy the ~~~~~
11
12 li $v0,5 #read a value
13 syscall #print the value
14 sw $v0,8($sp) # read store the local value on the stack
15 lw $t2,8($sp) #expr left hand is local
16 sw $t2,12($sp) #expr left hand store in sp
17 li $t3,10
18 lw $t2,12($sp) #lw the final local left hand
19 sgt $t2,$t2,$t3
20 sw $t2,12($sp)
21 lw $t0,12($sp) #expr is expr
22 beq $t0,0,LABEL1 #if the expression is false go to label1
23 li $a0,1
24 li $v0,1
25 syscall
26 j LABEL2 #go to label2
27
28 LABEL1 : #label1 is here
29
30 li $a0,0
31 li $v0,1
32 syscall
33
34 LABEL2 : #label2 is here
35
36 li $v0,0 # return zero
37 lw $ra ($sp) # get the original return address
38 lw $sp 4($sp) # get the old stack pointer
39 jr $ra #jump to the next instruction
```

MIPS OUTPUT:

```
King:spim yuhaolan$ vi test.asm
King:spim yuhaolan$ ./spim
Loaded: ./exceptions.s
(spim) load "test.asm"
(spim) run
11
1(spim) run
8
0(spim)
```

TEST TWO:

```
int x;

int main(int b)
{  int y;
   int sum;
   sum=0;
   x=1;
   read y;
   while (x <= y)
     {
     sum= sum + x;
     write x;
     x= x+1;
     }
   write sum;

}
```

the output:

```
1
2
3  main:
4  subu $sp,$sp,4 #substract the stack pointer for main to get space for global variables
5  addi $gp,$sp,0 #copy the space to global pointer for global variables
6  subu $sp, $sp,32 #substract the stack pointer for main
7  subu $t2,$sp,32 #get the space of function
8  sw $ra,($t2) #Store the return address
9  sw $sp,4($t2) #Store the old stack pointer
10 addu $sp,$t2,0 #copy the ~~~~~
11
12 li $t2,0 #load the number on the right to assign
13 sw $t2,16($sp) #store the number into the stack
14 li $t2,1 #load the number on the right to assign
15 sw $t2,0($gp) #store the number into the global pointer
16 li $v0,5 #read a value
17 syscall #print the value
18 sw $v0,12($sp) # read store the local value on the stack
19 LABEL3:
20
21 lw $t2,0($gp) #expr left hand is global
22 sw $t2,20($gp) #expr left hand store in gp
23 lw $t3,12($sp) #expr on right hand is local
24 lw $t2,20($gp) #lw the final global left hand
25 sle $t2,$t2,$t3
26 sw $t2,20($sp)
27 lw $t0,20($sp)
28 beq $t0,0,LABEL4
29 lw $t2,16($sp) #expr left hand is local
30 sw $t2,24($sp) #expr left hand store in sp
31 lw $t3,0($gp) #expr on right hand is global
32 lw $t2,24($sp) #lw the final local left hand
33 add $t2,$t2,$t3
34 sw $t2,24($sp)
35 lw $t2,24($sp) #load the expression on the left local
36 sw $t2,16($sp) #sw the expression on the left local value in assign
37 lw $a0,0($gp) #lw the global pointer value to print
38 li $v0,1
39 syscall
40 lw $t2,0($gp) #expr left hand is global
41 sw $t2,28($gp) #expr left hand store in gp
42 li $t3,1
43 lw $t2,28($gp) #lw the final global left hand
44 add $t2,$t2,$t3
45 sw $t2,28($sp)
46 lw $t2,28($sp) #load the expression on the left local
47 sw $t2,0($gp) #sw the expression on the left global value in assign
48 j LABEL3
49
50 LABEL4 :
51
52 lw $a0,16($sp) #lw the local pointer value to print
53 li $v0,1
54 syscall
55 li $v0,0 # return zero
56 lw $ra ($sp) # get the original return address
57 lw $sp 4($sp) # get the old stack pointer
58 jr $ra #jump to the next instruction
```

MIPS OUTPUT:

```
King:spim yuhaolan$ ./spim
Loaded: ./exceptions.s
(spim) load "test1.asm"
(spim) run
8
1234567836(spim) run
```

TEST THREE:

```
int y;
void main(void)
{    int x;
     write 5;
     read x;
     write x;
     read y;
     write x+y;
}
```

The MIPS CODE:

```
1
2
3  main:
4  subu $sp,$sp,4 #substract the stack pointer for main to get space for global variables
5  addi $gp,$sp,0 #copy the space to global pointer for global variables
6  subu $sp, $sp,16 #substract the stack pointer for main
7  subu $t2,$sp,16 #get the space of function
8  sw $ra,($t2) #Store the return address
9  sw $sp,4($t2) #Store the old stack pointer
10 addu $sp,$t2,0 #copy the ~~~~~
11
12 li $a0,5
13 li $v0,1
14 syscall
15 li $v0,5 #read a value
16 syscall #print the value
17 sw $v0,8($sp) # read store the local value on the stack
18 lw $a0,8($sp) #lw the local pointer value to print
19 li $v0,1
20 syscall
21 li $v0,5 #read a value
22 syscall #print the value
23 sw $v0,0($gp) # read store the global value on the stack
24 lw $t2,8($sp) #expr left hand is local
25 sw $t2,12($sp) #expr left hand store in sp
26 lw $t3,0($gp) #expr on right hand is global
27 lw $t2,12($sp) #lw the final local left hand
28 add $t2,$t2,$t3
29 sw $t2,12($sp)
30 lw $a0,12($sp)
31 li $v0,1
32 syscall
33 lw $t2,8($sp) #expr left hand is local
34 sw $t2,12($sp) #expr left hand store in sp
35 lw $t3,0($gp) #expr on right hand is global
36 lw $t2,12($sp) #lw the final local left hand
37 add $t2,$t2,$t3
38 sw $t2,12($sp)
39 li $v0,0 # return zero
40 lw $ra ($sp) # get the original return address
41 lw $sp 4($sp) # get the old stack pointer
42 jr $ra #jump to the next instruction
```

MIPS OUTPUT:

```
Loaded: ./exceptions.s
(spim) load "test2.asm"
(spim) run
59
98
17(spim)
```