BPOS: Excercises for week 7

student: Dimitri Tabatadze

May 7, 2023

Solutions

1. I have reformated the code to look more readable and more navigable:

```
1  int b;
2  int y;
3
4  int main() {
5      b = 5;
6      y = 11;
7      if b > 1 {
8          b = f1(b, b - 1)
9      };
10     return 0
11 };
12
13 int f1(int y, int z) {
14     while b > 0 {
15         if y > 12 {
16             b = (y - 1) * f1(y - 1, y)
17         } else {
18             if y > 9 {
19                 y = y - 1;
20                 b = 5
21             } else {
22                 b = 5;
23                 y = y - 1
24             }
25         }
26     };
27     y = y - 1;
28     return b
29 }
```

   There are a couple of problems with this code.

   1. Due to an invalid statement on line 16, the code won't compile. There is no production for that kind of a statement.
   2. Even though there are three subtrees with border words `y=y-1` (on lines 19, 23, 27), only one of them (line 23) will ever be reached.

   If we don't ignore problem 1, this task will not be completable. If we assume that all occurrences of `y=y-1` are reachable, the program rests would look like this:

   - After line 19: $c'.pr =$ `b=5;`$ft($`f1`$).body$`;return 0`
   - After line 23: $c'.pr = ft($`f1`$).body$`;return 0`
   - After line 27: $c'.pr =$ `return b;return 0`

   If we count computations where $hd(c.pr)$: `y=y-1` as **occurrences** of `y=y-1`, then after the first three (and in fact all infinite* occurences), $c'.pr = ft($`f1`$).body$`;return 0`

2. The expression to translate: `z + x[z]`

$$displ(\texttt{z}, \$gm) = size(vec) = (6 \cdot size(int))_{32} = (6 \cdot 4)_{32} = 24_{32}$$
$$displ(\texttt{x}, \$gm) = 0_{32}$$

The register `bpt`, where the pointer to the start of the region where global memory is contained, is the register number 28[1] which is called `$gp` in MIPS.

Following the steps for translating variable names[2], array elements[3] and binary arithmetic operations[4], we get the following MIPS code:

- Load `x[z]` into `$t0`

```
addi  $t0 bpt displ(x, $gm)
deref($t0)
addi  $t1 bpt displ(z, $gm)
deref($t1)
addi  23 0 size(int)
mul($t1, $t1, 23)
add   $t0 $t0 $t1
deref($t0)
```

- Load `z` into `$t1`

```
addi  $t1 bpt displ(z, $gm)
deref($t1)
```

- perform addition

```
add   $t0 $t0 $t1
```

After expanding the macros:

```
addi  $t0 $gp 0
lw    $t0 $t0 0
addi  $t1 $gp 24
lw    $t1 $t1 0
addi  $s7 $zero 4
mul   $t1 $t1 $s7
add   $t0 $t0 $t1
lw    $t0 $t0 0
addi  $t1 $gp 24
lw    $t1 $t1 0
add   $t0 $t0 $t1
```

[1] 12.1.1 Memory Map. page 255.
[2] 12.2.6 Variable Names. page 291-293.
[3] 12.2.8 Array Elements. page 294-295.
[4] 12.2.1 Binary Arithmetic Operations. page 298-299.

3. The expression to translate: `while z>0 { z=z-2 }`

$$displ(\texttt{z}, \$gm) = size(vec) = (6 \cdot size(int))_{32} = (6 \cdot 4)_{32} = 24_{32}$$
$$bw(n1) = \texttt{z>0}$$
$$bw(n3) = \texttt{z=z-2}$$

Following the steps for translating a while loop[5], comparison[6], assignment[7] and binary arithmetic operations[4], we get the following MIPS code:

- `code(n1)`

```
addi  $t0 bpt displ(z, $gm)
deref($t0)
addi  $t1 0 0
slt   $t0 $t0 $t1
```

$j = \$t0$

- `beqz j |code(n3)|+2`

```
beqz  $t0 5+2
```

- `code(n3)`

```
addi  $t0 bpt displ(z, $gm)
addi  $t1 bpt displ(z, $gm)
deref($t1)
addi  $t2 0 2
sub   $t1 $t1 $t2
sw    $t0 $t1 0
```

- `blez 0 -(|code(n1)|+|code(n3)|+1)`

```
blez  0 -(4+5+1)
```

After expanding the macros:

```
addi  $t0 $gp 24
lw    $t0 $t0 0
addi  $t1 $zero 0
slt   $t0 $t0 $t1
beqz  $t0 7
addi  $t0 $gp 24
addi  $t1 $gp 24
lw    $t1 $t1 0
addi  $t2 $zero 2
sub   $t1 $t1 $t2
sw    $t0 $t1 0
blez  $zero -10
```

[5] 12.3.3 While Loop. page 305-307.
[6] 12.2.13 Comparison. page 300.
[7] 12.3.1 Assignment. page 303-304.