BPOS: Excercises for week 7

student: Dimitri Tabatadze (group: G3)

May 9, 2023

Solutions

1. I have adjusted the whitespaces to make the code 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 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 equal to `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;while b>0{if y>12{b=(y-1)*f1(y-1,y)}else{if y>9{y=y-1;b=5}else{b=5;y=y-1}}};y=y-1;return b;return 0`
- After line 23: $c'.pr =$ `while b>0{if y>12{b=(y-1)*f1(y-1,y)}else{if y>9{y=y-1;b=5}else{b=5;y=y-1}}};y=y-1;return b;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), $c'.pr = ft(\texttt{f1}).body$`;return 0`

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

$$
\begin{aligned}
displ(\texttt{z}, \$gm) &= size(vec) = (6 \cdot size(int))_{32} = (6 \cdot 4)_{32} = 24_{32} \\
displ(\texttt{x}, \$gm) &= 0_{32} \\
size(int) &= 4 \\
enc(size(int), uint) &= enc(4, uint) = 4_{32} = 00000000000000000000000000000100
\end{aligned}
$$

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)
addi  $t1 bpt displ(z, $gm)
deref($t1)
gpr(23) = enc(size(int),uint)
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
```

Expanding the macros and replacing variables

- `gpr(k) = S` →
  - `lui k 0bS[31:16]`
  - `ori k 0bS[15:0]`

```
addi  $t0 bpt displ(x, $gm)
addi  $t1 bpt displ(z, $gm)
deref($t1)
lui 23 0000000000000000
ori 23 0000000000000100
mul($t1, $t1, 23)
add   $t0 $t0 $t1
deref($t0)
addi  $t1 bpt displ(z, $gm)
deref($t1)
add   $t0 $t0 $t1
```

- `bpt` → `$gp`
- `23` → `$s7`
- `0` → `$zero`

```
addi  $t0 $gp displ(x, $gm)
addi  $t1 $gp displ(z, $gm)
deref($t1)
lui   $s7 0000000000000000
ori   $s7 0000000000000100
mul($t1, $t1, $s7)
add   $t0 $t0 $t1
deref($t0)
addi  $t1 $gp displ(z, $gm)
deref($t1)
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.

- `displ(x, $gm)` → `0`
- `displ(z, $gm)` → `24`
  `size(int)` → `4`

```
addi  $t0 $gp 0
addi  $t1 $gp 24
deref($t1)
lui   $s7 0000000000000000
ori   $s7 0000000000000100
mul($t1, $t1, $s7)
add   $t0 $t0 $t1
deref($t0)
addi  $t1 $gp 24
deref($t1)
add   $t0 $t0 $t1
```

- `deref(j)` → `lw j j 0`

```
addi  $t0 $gp 0
addi  $t1 $gp 24
lw $t1 $t1 0
lui   $s7 0000000000000000
ori   $s7 0000000000000100
mul($t1, $t1, $s7)
add   $t0 $t0 $t1
lw $t0 $t0 0
addi  $t1 $gp 24
lw $t1 $t1 0
add   $t0 $t0 $t1
```

- Expanding software multiplication[5] is a little more complex than the previous expansions. I follow the steps turning to get:

`mul(k, i, j)` →

```
addi  24 0 1
addi  26 i 0
addi  27 0 0
and   25 j 24
beq   25 0 2
add   27 27 26
add   24 24 24
add   26 26 26
bne   24 0 -5
```

```
addi  $t0 $gp 0
addi  $t1 $gp 24
lw    $t1 $t1 0
lui   $s7 0000000000000000
ori   $s7 0000000000000100
addi 24 0 1
addi 26 $t1 0
addi 27 0 0
and 25 $s7 24
beq 25 0 2
add 27 27 26
add 24 24 24
add 26 26 26
bne 24 0 -5
addi $t1 27 0
add   $t0 $t0 $t1
lw    $t0 $t0 0
addi  $t1 $gp 24
lw    $t1 $t1 0
add   $t0 $t0 $t1
```

but here, we need to replace the numbers of the registers with the corresponding names:

[5] 9.2 Software Multiplication. page 147-148.

$0 \rightarrow$ \$zero

$24 \rightarrow$ \$t8

$25 \rightarrow$ \$t9

$26 \rightarrow$ \$k0

$27 \rightarrow$ \$k1

```
addi  $t0 $gp 0
addi  $t1 $gp 24
lw    $t1 $t1 0
lui   $s7 0000000000000000
ori   $s7 0000000000000100
addi  $t8 $zero 1
addi  $k0 $t1 0
addi  $k1 $zero 0
and   $t9 $s7 $t8
beq   $t9 $zero 2
add   $k1 $k1 $k0
add   $t8 $t8 $t8
add   $k0 $k0 $k0
bne   $t8 $zero -5
addi  $t1 $k1 0
add   $t0 $t0 $t1
lw    $t0 $t0 0
addi  $t1 $gp 24
lw    $t1 $t1 0
add   $t0 $t0 $t1
```

And finaly, after expanding the macros and replacing the variables with their values we get the following MIPS code:

```
addi  $t0 $gp 0
addi  $t1 $gp 24
lw    $t1 $t1 0
addi  $s7 $zero 4
addi  $t8 $zero 1
addi  $k0 $t1 0
addi  $k1 $zero 0
and   $t9 $s7 $t8
beq   $t9 $zero 2
add   $k1 $k1 $k0
add   $t8 $t8 $t8
add   $k0 $k0 $k0
bne   $t8 $zero -5
addi  $t1 $k1 0
add   $t0 $t0 $t1
lw    $t0 $t0 0
addi  $t1 $gp 24
lw    $t1 $t1 0
add   $t0 $t0 $t1
```

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

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

Following the steps for translating a while loop[6], comparison[7], assignment[8] and binary arithmetic operations[4], we get the following MIPS code:

- `code(n1)`

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

$j = \$t0$

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

```
beqz  $t0 7+2
```

- `code(n3)`

  The code for an assignment and an unary operation (with $\circ = -$)

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

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

```
blez  0 -(4+7+1)
```

I have ommited the steps (since it's basicaly the same as it was in the previous task), but after expanding the macros and replacing the variables with their values we get the following MIPS code:

```
addi  $t0 $gp 24
lw    $t0 $t0 0
lui   $t1 0000000000000000
lui   $t1 0000000000000000
slt   $t0 $t1 $t0
beqz  $t0 9
addi  $t0 $gp 24
addi  $t1 $gp 24
lw    $t1 $t1 0
lui   $t2 0000000000000000
ori   $t2 0000000000000010
sub   $t1 $t1 $t2
sw    $t0 $t1 0
blez  $zero -12
```

[6] 12.3.3 While Loop. page 305-307.
[7] 12.2.13 Comparison. page 300.
[8] 12.3.1 Assignment. page 303-304.