BPOS: Excercises for week 8

student: Dimitri Tabatadze (group: G3)

May 16, 2023

Solutions

1. (a)

$$\begin{aligned}
ba(a, c^0) &= sbase \\
ba(b, c^0) &= sbase + 4 \\
va(a, c^1) &= 256 \\
d^0.m(lv(a,c)) &= enc(va(a, c^1), int) = enc(256, int) \\
d^1.gpr(1) &= enc(va(a, c^1), int) = enc(256, int) \\
d^2.gpr(2) &= ba(b, c^1) = sbase + 4 = d^1.gpr(28) + 4 && \text{without affecting register 1.} \\
d^3.gpr(1) &= 0^4 d^2.gpr(1)[31:4] = enc(16, int) && \text{code line number 11.} \\
d.m(d^3.gpr(2)) &= d^3.gpr(1) = enc(16, int) && (d, c) \text{ is the desired configuration.} \\
enc(va(b,c), int) &= d.m(d^3.gpr(2)) = enc(16, int) \\
\implies va(b, c) &= 16
\end{aligned}$$

(b) no, since register 1 might get used by the statement

```
      gpr(2) = b&
```

and the value 256 (which later becomes 16 because of the `srl` instruction in the `asm` block) would get lost.

2.

$$\begin{aligned}
\langle Reg \rangle &\rightarrow \texttt{0} \mid \texttt{1} \mid \ldots \mid \texttt{30} \mid \texttt{31} \\
\langle RegS \rangle &\rightarrow \langle Reg \rangle \mid \langle Reg \rangle\texttt{,}\langle RegS \rangle && \text{(we could use } \texttt{;} \text{ instead of } \texttt{,}\text{)} \\
\langle AsmSt \rangle &\rightarrow \texttt{gpr(}\langle Reg \rangle\texttt{) = } \langle E \rangle \texttt{ \{}\langle RegS \rangle\texttt{\}} \mid \\
&\qquad \langle id \rangle \texttt{ = gpr(}\langle Reg \rangle\texttt{) \{}\langle RegS \rangle\texttt{\}}
\end{aligned}$$

3. From **Lemma 93** I use only

$$d^k.gpr(j_5) = \begin{cases} ba(va(e,c), c), & R(i) = 1 \wedge pointer(t) \wedge va(e,c) \neq null, \\ enc(va(e,c), t), & R(i) = 1 \wedge t \in ET \end{cases}$$

since $R(n') = 1$ is given where n' is the node of the expression e in the derivation tree. Let j' be the pebble with the value of n'.

$$d''.gpr(j') = \begin{cases} ba(va(e,c), c), & R(n') = 1 \wedge pointer(t) \wedge va(e,c) \neq null, \\ enc(va(e,c), t), & R(n') = 1 \wedge t \in ET \end{cases}$$

Since the last instruction is `addi j j' 0`, $d''.gpr(j) = d''.gpr(j')$, so

$$d''.gpr(j) = \begin{cases} ba(va(e,c), c), & pointer(t) \wedge va(e,c) \neq null, \\ enc(va(e,c), t), & t \in ET \end{cases}$$

Lemma 94:

$$d \rightarrow^*_{code(n,J)} d' \wedge j \in J \rightarrow d.gpr(j_5) = d'.gpr(j_5)$$

where

$$J \subset [1:27]$$

Let's prove that $i \in J \rightarrow d.gpr(i) = d''.gpr(i)$. $d \rightarrow_{code(n,J)} d''$ where $code(n, J)$ is

```
code(n', J)
addi j j' 0
```

It's obvious that $j \notin J$ and given that $code(n', J)$ won't affect J (from **Lemma 94**) and the last instruction of $code(n, J)$, `addi j j' 0` won't change anything but the register j so the set J will remain unchanged.

Lemma 127 suggests that the registers $i \in [28 : 31]$ always remain unchanged. However the syntax for C+A doesn't forbid us from accessing the those registers.

4. (a)

```
int a;

int main() {
  a = 5;
  gpr(1) = a&;
  while (a > 0) {
    asm(
      addi  1 1 -1
    )
  };
  return 0
}
```

(b) Only 1 step is needed. After one step, the whole while block will be loaded — the start of the node of the while loop will be the 2nd next step

(c)

```
int a;

int main() {
  a = 5;
  gpr(1) = a &;
  while (a > 0) {
    if (a != 10) {
      asm(
        addi  1 1 -1
      )
    } else {
      a = 10
    }
  };
  return 0
}
```