PCC Exercise 1

- On page 2 is an example program in machine code.
- Pages 3-5 define a Verification Condition Generator.
- Page 6 shows the outline of the computation of the verification condition of the example program and asks you to compute one of its conjuncts.
- Informally, what is the meaning of the resulting formula? What must be proved and why?

Compiled Program with Hints

Precondition: $r_0 :_m intlist \wedge r_3 = 0$

	`ADD` $r_1 := r_3 + r_3$	`%initialize total to 0`
	`INV` $r_0 :_m intlist \wedge r_1 :_m int \wedge r_3 = 0$	
L1	`LD` $r_5 := m(r_0 + 0)$	`%`r_5 `gets list tag`
	`BEQ` $(r_5 = r_3)$ *L2*	`%jump if list tag is 0`
	`LD` $r_2 := m(r_0 + 1)$	`%load next int in` r_2
	`LD` $r_0 := m(r_0 + 2)$	`%`r_0 `gets pointer to rest`
	`ADD` $r_1 := r_1 + r_2$	`%add next int to total`
	`BEQ` $(r_3 = r_3)$ *L1*	`%jump back`
	`INV` $r_1 :_m int$	
L2	`ADDC` $r_0 := r_1 + 0$	`%put total in` r_0
	`RET`	

Definition of Verification Condition Generator

- Let Π be the list of instructions output by the certifying compiler. Let Π_i be the instruction at position i in Π.
- Note: VC_{i+1} is needed to compute VC_i.

.

$$VC_i = \begin{cases} [(r_{s1}+r_{s2})/r_d]VC_{i+1} & \text{if } \Pi_i \text{ is } \mathtt{ADD\ r_d{:=}r_{s1}{+}r_{s2}} \\ [(r_s+c)/r_d]VC_{i+1} & \text{if } \Pi_i \text{ is } \mathtt{ADDC\ r_d{:=}r_s{+}c} \\ [m(r_s+c)/r_d]\ VC_{i+1} \wedge readable(r_s+c) & \\ & \text{if } \Pi_i \text{ is } \mathtt{LD\ r_d{:=}m(r_s{+}c)} \\ [upd(m,r_{s2}+c,r_{s1})/m]\ VC_{i+1} \wedge writable(r_{s2}+c) & \\ & \text{if } \Pi_i \text{ is } \mathtt{ST\ m(r_{s2}{+}c){:=}r_{s1}} \end{cases}$$

Definition of VCG (continued)

$$VC_i = \begin{cases} (r_{s1} = r_{s2} \Rightarrow VC_{i+c-1}) \wedge (\neg(r_{s1} = r_{s2}) \Rightarrow VC_{i+1}) & \\ & \text{if } \Pi_i \text{ is } \mathtt{BEQ\ (r_{s1}{=}r_{s2})\ c} \\ (r_{s1} > r_{s2} \Rightarrow VC_{i+c-1}) \wedge (\neg(r_{s1} > r_{s2}) \Rightarrow VC_{i+1}) & \\ & \text{if } \Pi_i \text{ is } \mathtt{BGT\ (r_{s1}{>}r_{s2})\ c} \\ post & \text{if } \Pi_i \text{ is } \mathtt{RET} \\ p & \text{if } \Pi_i \text{ is } \mathtt{INV}\ p \end{cases}$$

- *post* is the postcondition.
- Every jump point must be proceeded by an `INV` statement.

Verification Condition

- Let *Inv* be the set of line numbers containing `INV` machine instructions. Also, $0 \in Inv$.
- Inv_0 is the precondition.
- Inv_i denotes the formula at line *i*.
- *SP* is the function computing the safety predicate (verification condition) from the code.

$$SP(\Pi, Inv, post) = \forall k\ \forall r_k \bigwedge_{i \in Inv} Inv_i \Rightarrow VC_{i+1}$$

VCGen Applied to Example Program

`0:` $r_0 :_m intlist \wedge r_3 = 0$

$\vdots$

`2: INV` $r_0 :_m intlist \wedge r_1 :_m int \wedge r_3 = 0$

$\vdots$

`9: INV` $r_1 :_m int$

$$(Inv_0 \Rightarrow VC_1) \wedge (Inv_2 \Rightarrow VC_3) \wedge (Inv_9 \Rightarrow VC_{10})$$

- Exercise: Compute $(Inv_2 \Rightarrow VC_3)$