Sheet 9

Digdarshan Kunwar

November 2018

Problem 9.1

a)

#	Machine Code	Assembly Code	Description
0	001 1 0001	LOAD #1	Load the value 1 in accumulator
1	010 0 1111	STORE 15	Store the value of the accumulator in memory location 15
2	001 1 0000	LOAD #0	Load the value 0 in accumulator
3	101 1 0100	EQUAL#4	Skip the next instruction if the accumulator equals to 4
4	110 1 0110	JUMP #6	Jump to instruction 6 (set program counter to 6)
5	111 1 0000	HALT	Stop execution
6	001 0 0011	LOAD 3	Load the value of memory location 3 into the accumulator
7	100 1 0001	SUB #1	Subtract the value 1 from the accumulator
8	010 0 0011	STORE 3	Store the value of accumulator in memory location 3
9	001 0 1111	LOAD 15	Load the value of memory location 15
10	011 0 1111	ADD 15	Add the value of memory location 15 to the accumulator
11	010 0 1111	STORE 15	Store the value of accumulator in memory location 15
12	110 1 0010	JUMP #2	Jump to instruction 2 (Set program counter to 2)
13	000 0 0000	-	no instruction / data, initialized to 0
14	000 0 0000	-	no instruction / data, initialized to 0
15	000 0 0000	-	no instruction / data, initialized to 0

b)

Here, the as program runs

1. The program subtracts the value of memory location 3 by 1
2. The programs initially stores value 1 in memory location 15
3. The programs doubles the value stored in memory location 15
4. Jumps back follow the same instructions and loops until the value in memory address 3 is 0. Then Halts

Here as the program modifies the the value of the memory location 3 by -1 everytime the compare is false.
So the value in the memory address is intially 1 which gets doubled. The value in memory location 15 is doubled until the value in memory location 3 is 0.
So it is doubled 4 times ie ((((1*2)*2)*2)*2) = 16
Here the value 16 is 10000 which is overflow for the 4 bit storage.
So the final value is 0 in memory address 15.

Problem 9.2

a)

Here we know that,
op is associative : (x op y) op z = x op (y op z)
e is neutral element: e op x = x and x op e = x
Now we know that,

To prove :
`foldl op e xs = foldr op e xs`

The Base Conditions are :
foldr op e [ ] = e
foldl op e [ ] = e
foldr op e [ ] = foldl op e [ ]

Here,
xs is a finite list.
e is the neutral element.

The Induction Hypothesis

$$foldl\ op\ e\ xs\ = foldr\ op\ e\ xs$$

Adding an extra element x in list xs.
Inductive step:

$$\begin{aligned} foldl\ op\ e\ (x:xs) &= (op\ e\ x)\ `op`\ (foldl\ op\ e\ xs) \\ &= \ x\ `op`\ (foldl\ op\ e\ xs) \end{aligned}$$

$$foldr\ op\ e\ (x:xs) = x\ `op`\ (foldr\ op\ e\ xs)$$

$$\begin{aligned} foldl\ op\ e\ (x:xs) &= x\ `op`\ (foldl\ op\ e\ xs) \\ &= x\ `op`\ (foldr\ op\ e\ xs) && \texttt{From Induction Hypothesis} \\ &= foldr\ op\ e\ (x:xs) \\ &= LHS\ Proved \end{aligned}$$

---

b)

To prove :
`foldr op1 e xs = foldl op2 e xs`

Given,
x ‘op1‘ (y ‘op2‘ z) = (x ‘op1‘ y) ‘op2‘ z
x ‘op1‘ e = e ‘op2‘ x

The Base Conditions are :
foldr op1 e [ ] = e
foldl op2 e [ ] = e

foldr op1 e [ ] = foldl op2 e [ ]

Here,
xs is a finite list.
The Induction Hypothesis

$$foldr\ op1\ e\ xs\ = foldl\ op2\ e\ xs$$

Adding an extra element x in list xs.
Inductive step:

$$\begin{aligned}
LHS &= foldr\ op1\ e\ (x : xs) \\
RHS &= \ foldl\ op2\ e\ (x : xs) \\
&= \ (foldl\ op2\ (op2\ e\ x)\ xs) \\
&= \ (foldl\ op2\ (op1\ x\ e)\ xs) \\
&= \ x\ `op1`\ (foldl\ op2\ e\ xs) \\
&= \ x\ `op1`\ (foldr\ op1\ e\ xs) && \texttt{By Induction Hypothesis} \\
&= \ foldr\ op1\ e\ (x : xs) \\
&= LHS\ Proved
\end{aligned}$$

---

c)

The Base Condition :
`foldr op a [] = foldl op’ a (reverse [])= a`

Here,
xs is a finite list.
The Induction Hypothesis

$$foldr\ op\ a\ xs\ \ = \ \ foldl\ op'\ a\ (reverse\ xs)$$

Adding an extra element x in list xs.
Inductive step:

$$\begin{aligned}
LHS &= foldr\ op\ a\ (x : xs) \\
RHS &= \ foldl\ op'\ a\ (reverse\ (x : xs)) \\
&= \ foldl\ op'\ a\ ((reverse\ xs) :\ x) \\
&= \ (foldl\ op'\ a\ (reverse\ xs))\ `op'`\ x \\
&= \ (foldr\ op\ a\ xs)\ `op'`\ x && \texttt{From Induction Hypothesis} \\
&= \ x\ `op`\ (foldr\ op\ a\ xs) \\
&= \ foldr\ op\ a\ (x : xs) \\
&= \ LHS\ Proved
\end{aligned}$$