CSE 232 SPRING 2020

FINAL PROJECT

ABDULLAH CELIK

171044002

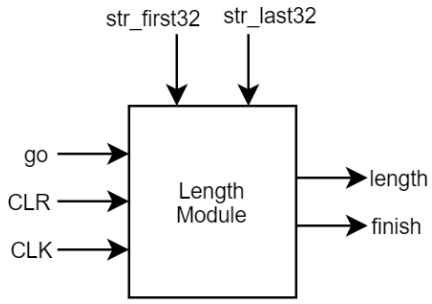
C Code I Implemented:

```
#include <stdio.h>

int LengthModule(char[]);
int InputChecker(char str[], char foundedStr[], int length, char input);

int main()
{
   char input;
   char str[8] = "hello";
   char foundedStr[8] = "________";
   int founded = 0;
   int error = 0;
   int returned = 0;
   int length = LengthModule(str);

   while(founded < length && error != 10)
   {
      printf("Input: ");
      scanf(" %c",&input);
      if((returned = InputChecker(str,foundedStr,length,input)) > 0)
         founded += returned;
      else
         ++error;

      printf("%s\n",foundedStr);
   }

   printf("Word: %s\n",str);
   return 0;
}

int LengthModule(char str[])
{
   int length=0;

   while(str[length++]);

   return length-1;
}

int InputChecker(char str[], char foundedStr[], int length, char input)
{
   int returned = 0;
   int i=0;

   while(i < length)
   {
      if(str[i] == input)
      {
         foundedStr[i] = input;
         ++returned;
```

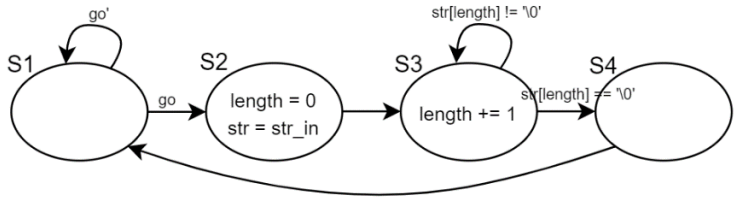
```
        }
        ++i;
      }

      return returned;
    }
```

Length Module

- **Overview**

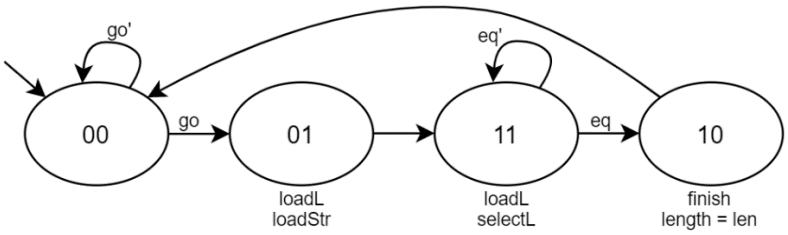


- **ASM**



- **State Diagram**

Inputs: go, eq, len
Outputs: loadL, selectL, loadStr, length



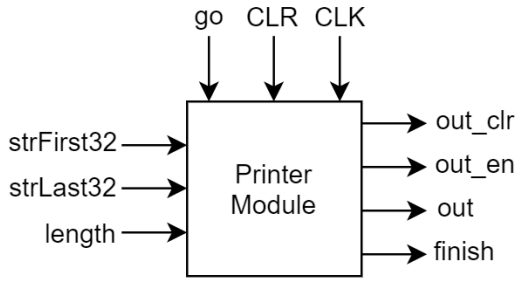
- **Truth Table and Boolean Expresions**

Inputs					Outputs						
s1	s0	eq	go	len	n1	n0	loadL	selectL	loadStr	finish	length
0	0	X	0	X	0	0	0	0	0	0	0
0	0	X	1	X	0	1	0	0	0	0	0
0	1	X	X	X	1	1	1	0	1	0	0
1	1	0	X	X	1	1	1	1	0	0	0
1	1	1	X	X	1	0	1	1	0	0	0
1	0	X	X	X	0	0	0	0	0	1	len

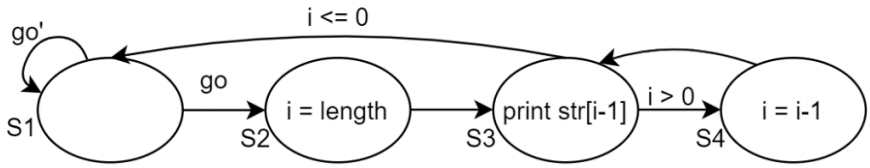
n1 = s0
n0 = s1'go + s1's0 + s0eq'
loadL = s0
selectL = s1s0
loadStr = s1's0
finish = s1s0'
length = s1s0’len

Printer Module

- **Overview**

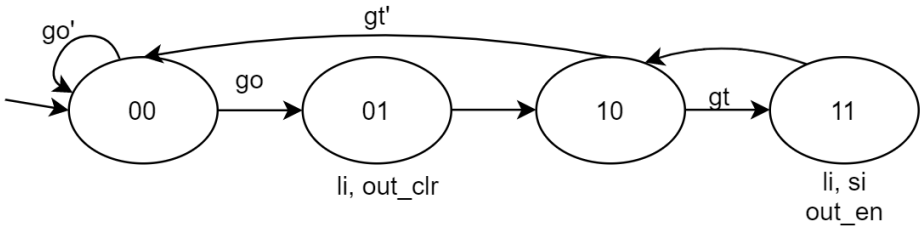


- **ASM**



- **State Diagram**

Inputs: go, gt
Outputs: li, si, out_clr, out_en, finish(in state S3 if i <= 0)



- **Truth Table and Boolean Expresions**

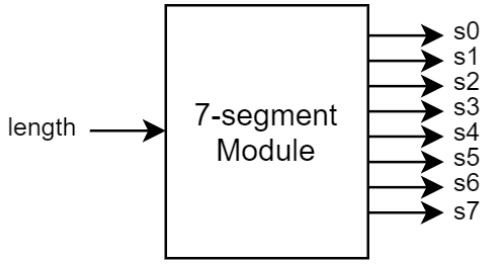
Inputs				Outputs						
s1	s0	gt	go	n1	n0	li	si	out_clr	out_en	finish
0	0	X	0	0	0	0	0	0	0	0
0	0	X	1	0	1	0	0	0	0	0
0	1	X	X	1	0	1	0	1	0	0
1	0	0	X	0	0	0	0	0	0	1
1	0	1	X	1	1	0	0	0	0	0
1	1	X	X	1	0	1	1	0	1	0

n1 = s1's0 + s1s0'gt + s1s0
n1 = s0(s1'+s1)+s1s0'gt
n1 = s0 + s1s0'gt
n1 = s0 + s1gt
n0 = s1's0'go + s1s0'gt
li = s1's0 + s1s0 = s0
si = s1s0
out_clr = s1's0
out_en = s1s0
finish = s1s0'gt'

7-segment Module

- **Overview**

7-segment Module is a combinatiaonal logic.



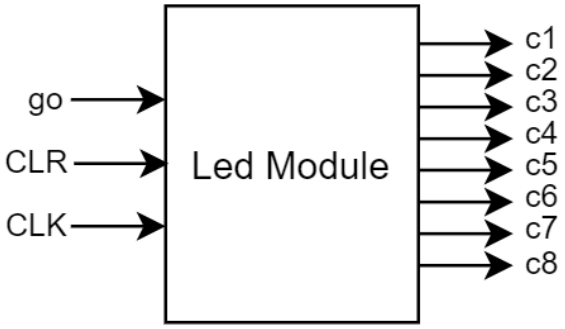
- **Truth Table and Boolean Expresions**

Inputs				Outputs							
I3	I2	I1	I0	s7	s6	s5	s4	s3	s2	s1	s0
0	0	0	0	0	1	1	1	0	1	1	1
0	0	0	1	0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	1	1	1	0	1
0	0	1	1	0	1	1	0	1	1	0	1
0	1	0	0	0	0	1	0	1	1	1	0
0	1	0	1	0	1	1	0	1	0	1	1
0	1	1	0	0	1	1	1	1	0	1	1
0	1	1	1	0	0	1	0	0	1	0	1
1	0	0	0	0	1	1	1	1	1	1	1
1	0	0	1	0	1	1	0	1	1	1	1
1	0	1	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0

s7 = 0
s6 = I3’I2’I0’ + I3’I2’I1 + I3’I1’I0 + I3’I1I0’ + I3I2’I1’
s5 = I2’I1’ + I3’I0 + I3’I2
s4 = I2’I1’I0’ + I3’I1I0’
s3 = I3’I2’I1 + I3’I2I1’ + I3’I1I0’ + I3I2’I1’
s2 = I3’I2’ + I3’I1’I0’ + I3’I1I0 + I2’I1’
s1 = I3’I1’I0’ + I3’I2I1’ + I3’I2I0’ + I3I2’I1’
s0 = I3’I2’I0’ + I3’I1 + I3’I2I0 + I3I2’I1’

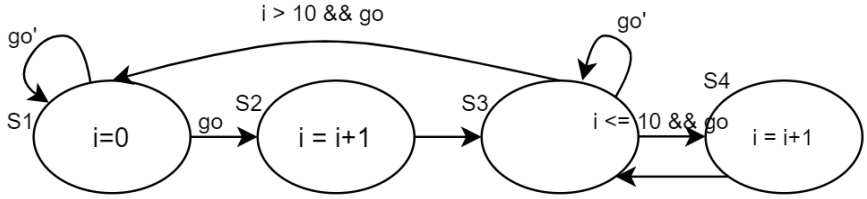
Led Module

- **Overview**



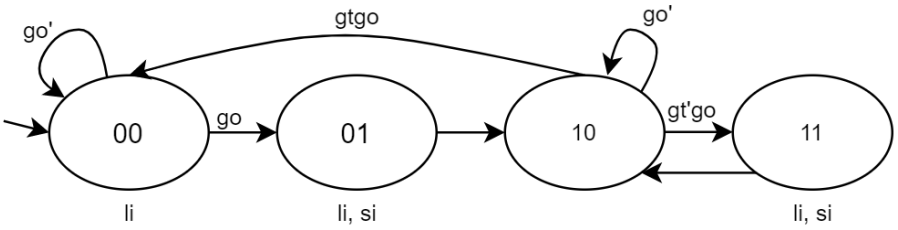
cX(X. column of display)

- **ASM**



- **State Diagram**

Inputs: go, gt
Outputs: li, si



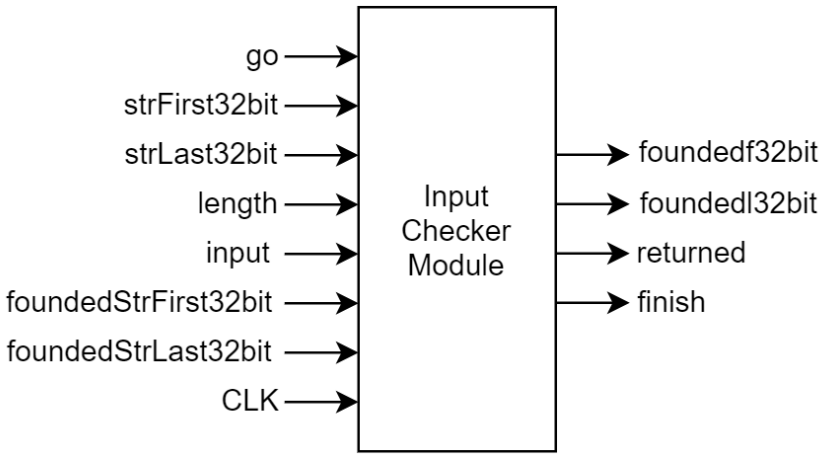
- **Truth Table and Boolean Expresions**

Inputs				Outputs			
s1	s0	gt	go	n1	n0	li	si
0	0	X	0	0	0	1	0
0	0	X	1	0	1	1	0
0	1	X	X	1	0	1	1
1	0	0	0	1	0	0	0
1	0	0	1	1	1	0	0
1	0	1	0	1	0	0	0
1	0	1	1	0	0	0	0
1	1	X	X	1	0	1	1

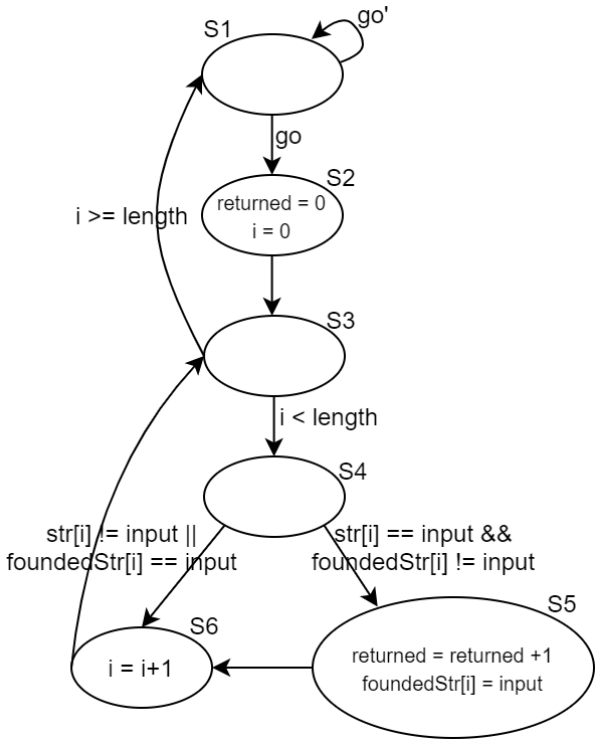
n1 = s0 + s1s0'gt' + s1s0'gtgo'
n1 = s0 + s1s0'(gt' + gtgo')
n1 = s0 + s1s0'gt' + s1s0'go'
n0 = s1's0'go + s1s0'gt'go
n0 = s0'go(s1' + s1gt')
n0 = s1's0'go + s0'gt'go
li = s1's0' + s0
si = s0

Input Checker Module

- **Overview**

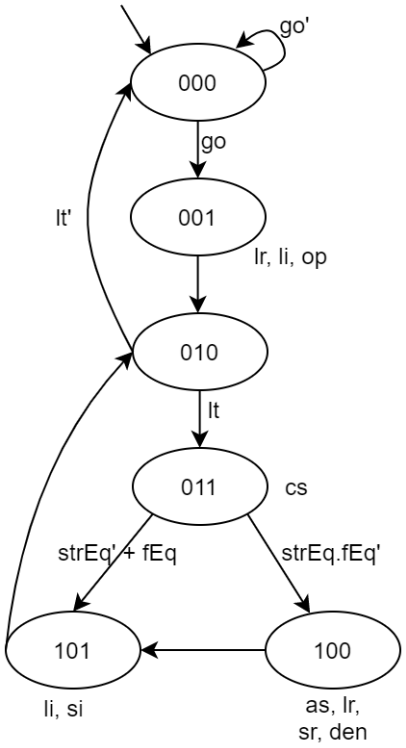


- **ASM**



- **State Diagram**

Inputs: go, strEq, fEq, lt
Outputs: li, si, lr, sr, cs, as, den, op, finish(in state 001 if lt is 0)



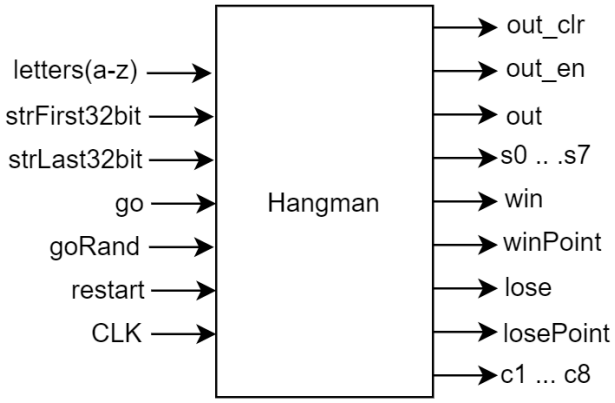
- **Truth Table and Boolean Expressions**

Inputs							Outputs										
s2	s1	s0	strEq	fEq	lt	go	n2	n1	n0	li	si	lr	sr	cs	as	den	op	finish
0	0	0	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	X	X	X	1	0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	X	X	X	X	0	1	0	1	0	1	0	0	0	0	1	0
0	1	0	X	X	0	X	0	0	0	0	0	0	0	0	0	0	0	1
0	1	0	X	X	1	X	0	1	1	0	0	0	0	0	0	0	0	0
0	1	1	0	0	X	X	1	0	1	0	0	0	0	1	0	0	0	0
0	1	1	0	1	X	X	1	0	1	0	0	0	0	1	0	0	0	0
0	1	1	1	0	X	X	1	0	0	0	0	0	0	1	0	0	0	0
0	1	1	1	1	X	X	1	0	1	0	0	0	0	1	0	0	0	0
1	0	0	X	X	X	X	1	0	1	0	0	1	1	0	1	1	0	0
1	0	1	X	X	X	X	0	1	0	1	1	0	0	0	0	0	0	0

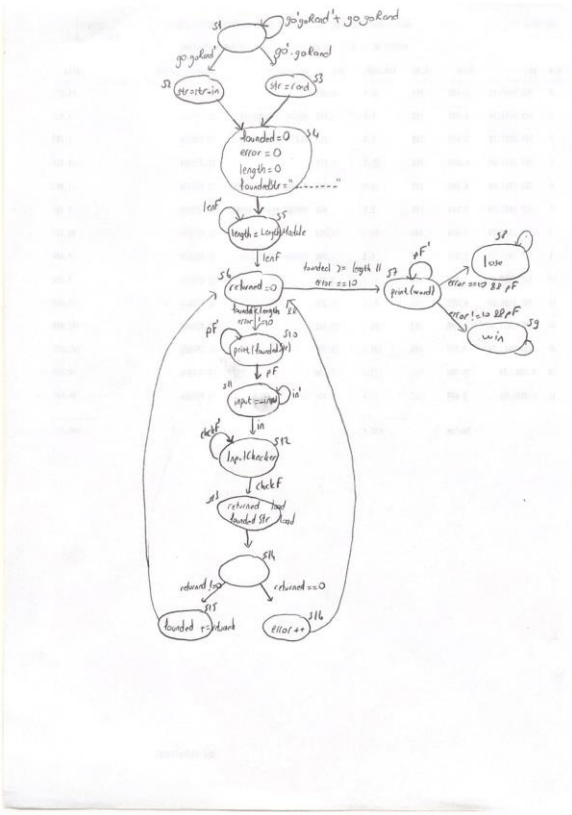
n2 = s2's1s0 + s2s1's0'
n1 = s1's0 + s2's1s0'lt
n0 = s2's1's0'go + s2's1s0'lt + s2's1s0strEq' + s2's1s0fEq + s2s1's0'
li = s1's0
si = s2s1's0
lr = s1'(s2 XOR s0)
sr = s2s1's0'
cs = s2's1s0
as = s2s1's0'
den = s2s1's0'(same with as)
op = s2's1's0
finish = s2's1s0'lt'

Hangman

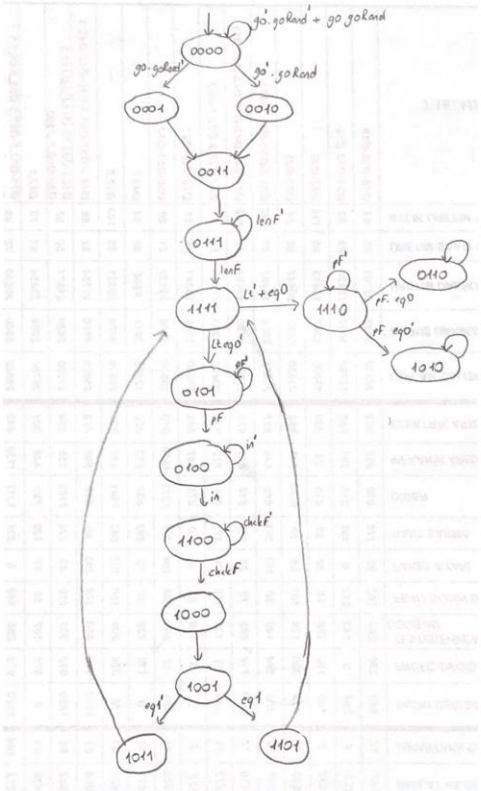
- **Overview**



- **ASM**



- **State Diagram**



- **Truth Table and Boolean expresions**

  *Attached zip as png

  n3 = s3's2s1s0lenF + s2s1's0'in + s3eq0'lt' + s3eq0pF' + s3s0eq0 + s3s0'eq0' + s3s2' + s3s1'
  n2 = s3s0eq1 + s3's2 + s2s1eq0 + s2s1pF' + s2s1'chckF' + s2s0 + s1s0
  n1 = s3s2s1's0 + s3's2's0 + s3's2'go'goRand + s3's1 + s2's1 + s2's0eq1' + s1s0' + s1eq0 + s1lt'
  rL = s3’s2’s1s0’
  strL = s3’s2’(s1 XOR s0)
  strS = s3’s2’s1s0’(same with rL)
  foundedL = s3’s2’s1s0 + s3s2’s1s0 = s2’s1s0
  foundedS = s3s2’s1s0
  returnedL = s3s2s1s0 + s3s2’s1’s0’
  returnedS = s3s2’s1’s0’
  errorL = s3’s2’s1s0 + s3s2s1’s0
  errorS = s3s2s1’s0
  err = s3s2s1’s0(same with errorS)
  lengthL = s3’s2’s1s0 + s3’s2s1s0 = s3’s1s0
  lengthS = s3’s2s1s0
  foundedStrL = s3’s2’s1s0 + s3s2’s1’s0’
  foundedStrS = s3s2’s1’s0’(same with returnedS)
  inL = s3’s2s1’s0’
  goL = s3’s2s1s0(same with lengthS)
  goP = s3s2s1s0’ + s3’s2s1’s0
  pS = s3s2s1s0’
  goChck = s3s2s1’s0’
  win = s3s2’s1s0’
  lose = s3’s2s1s0’
  CLR = s3’s2’s1’s0’