Exercise 1

Inputs: byte a[256],
      byte b, bit go

Outputs: byte freq
      bit done

State transition diagram:

- Wait state with self-loop labeled !go
- Wait → S0 on go
- S0: done := 0, i := 0, freq := 0
- S0 → S1
- S1 → S2 on i < 256
- S1 → S5 on (i < 256)'
- S2 → S3 on a[i] == b
- S2 → S4 on (a[i] == b)'
- S3: freq := freq + 1
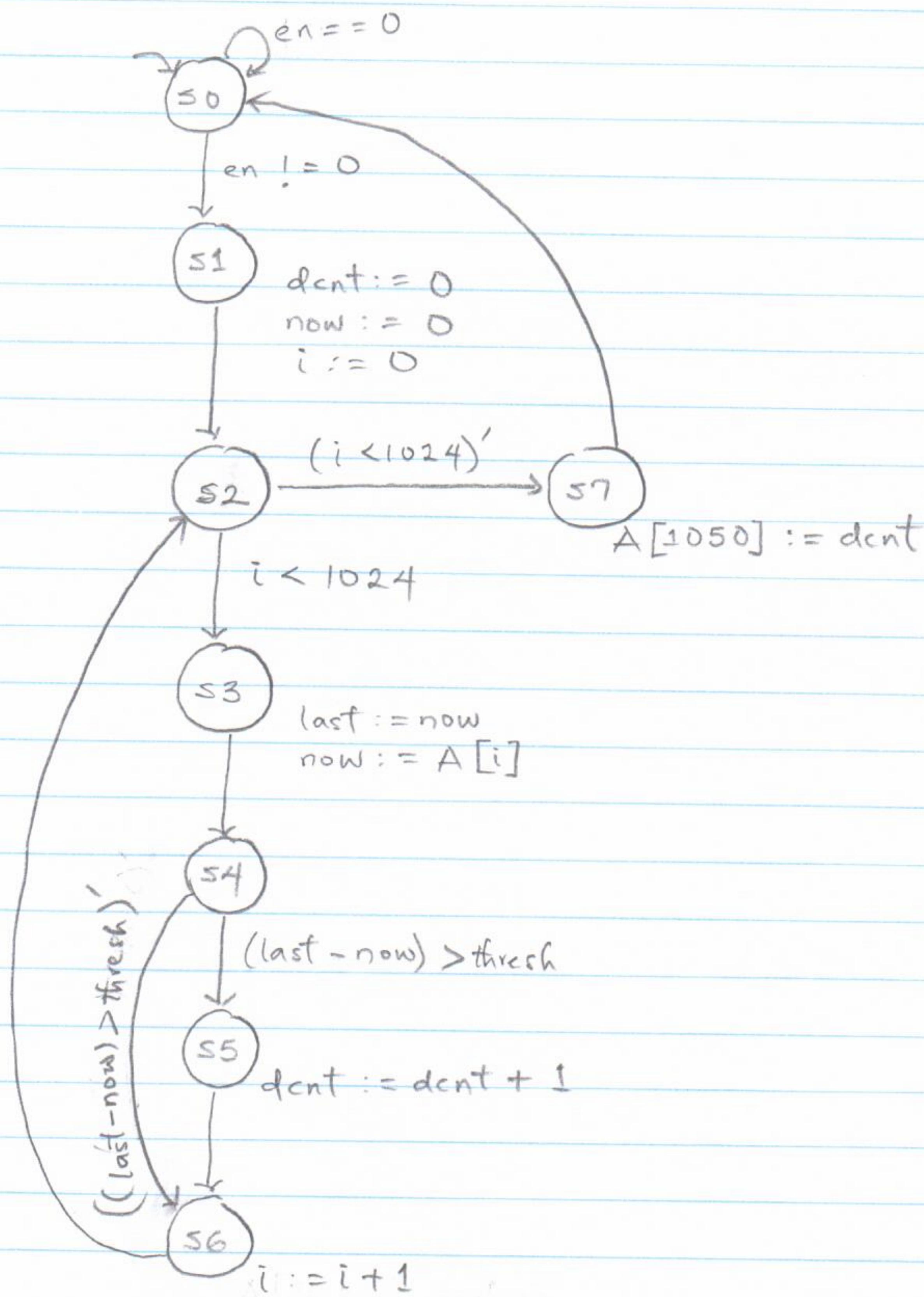- S3 → S4
- S4: i = i + 1
- S4 → S1
- S5: done := 1
- S5 → Wait
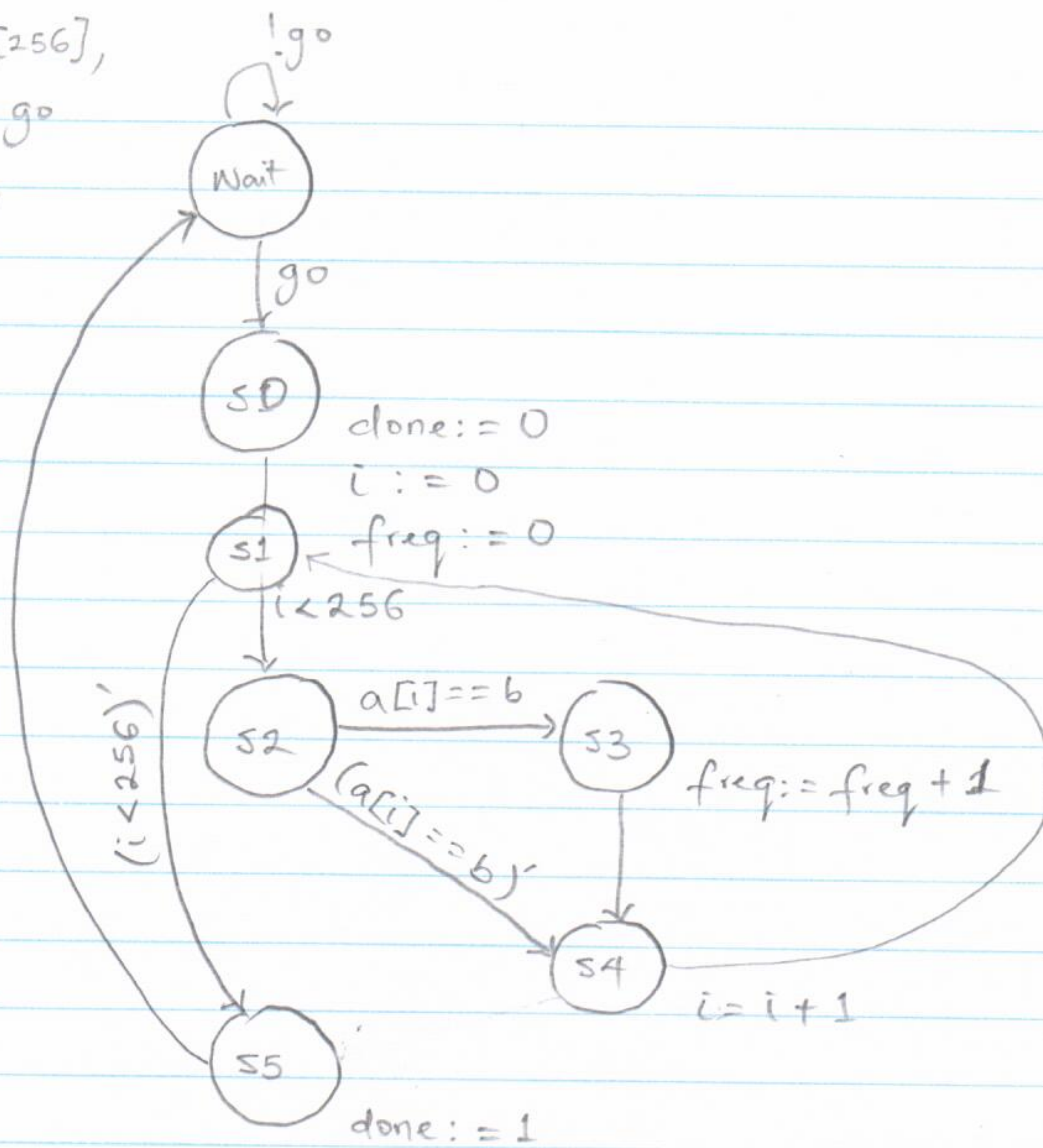
Exercise 2

input: en (8 bits), A[i] (32 bits), thresh (32 bits)
output: A[i] (32 bits)
local reg: i (16 bits), dcnt (16 bits), now (32 bits), last (32 bits)

S0 → (en == 0) self loop

S0 → S1 : en != 0

S1 → S2 : dcnt := 0, now := 0, i := 0

S2 → S7 : (i < 1024)'

S7 : A[1050] := dcnt

S2 → S3 : i < 1024

S3 → S4 : last := now, now := A[i]

S4 → S5 : (last - now) > thresh

S5 → S6 : dcnt := dcnt + 1

S4 → S6 : ((last - now) > thresh)'

S6 → S2 : i := i + 1

S7 → S0

Exercise 3

!start

wait → start → init

init: $i := 0$
$mod4\_count := 0$

wait ← !($i < 128$) ← while

while → $i < 128$ → init data

init data: $data := A[i]$

if

if → $data \% 4 == 0$ → inc_mod

if → !($data \% 4 == 0$) → inc_i

inc_mod: $mod4\_count = mod4\_count + 1$

inc_mod → inc_i

inc_i: $i = i + 1$

inc_i → while