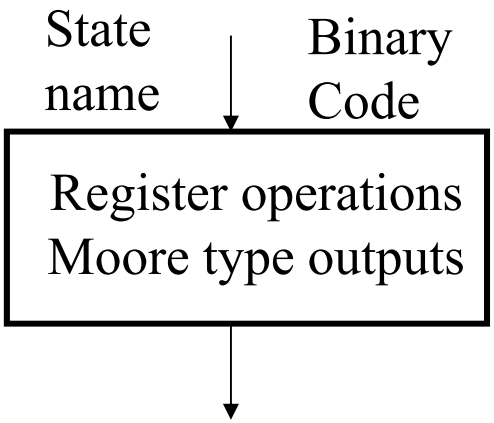
Algorithmic State Machines (ASM)

- ASM Charts
- Algorithmic State Machine with Datapath (ASMD)
- Timing
- Implementation

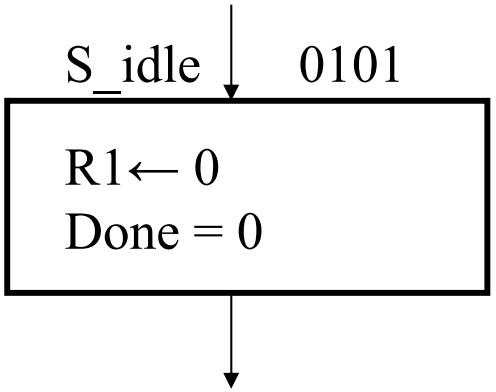
- Top-Down Design
  - Develop specification for problem
  - Split into smaller sub-problems
  - Repeat until sub-problems can be implemented
- Digital Systems
  - Separate controller from controlled hardware
  - Develop an overall architecture
  - Store data in flip-flops, registers and memory
  - Process data with combinational circuits
- Algorithmic State Machine and Datapath
  - ASM chart describes the controller
  - Annotated with datapath operations

State box: This represents a state in the sequence. Its name is on the top left and its code on the top right. Outputs corresponding to the state are listed in the box. These are unconditional and completed during one clock cycle. (Moore type output)

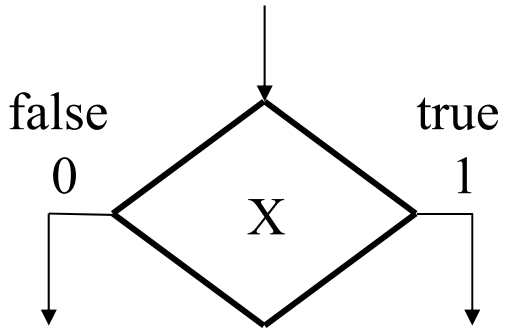


Example:

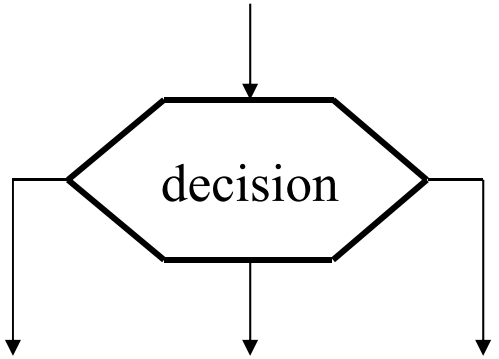
- State name: S_idle
- Binary encoding: 0101
- Register operation: R1 ← 0
  Register R1 is to be cleared to 0 during the next clock cycle
- Output signal asserted: Done = 0
  Control signal for datapath



Decision box: This allows a two way branch based on the condition written in the box. There is one exit path for true and one for false. If binary values are assigned, the paths are indicated by 0 and 1.

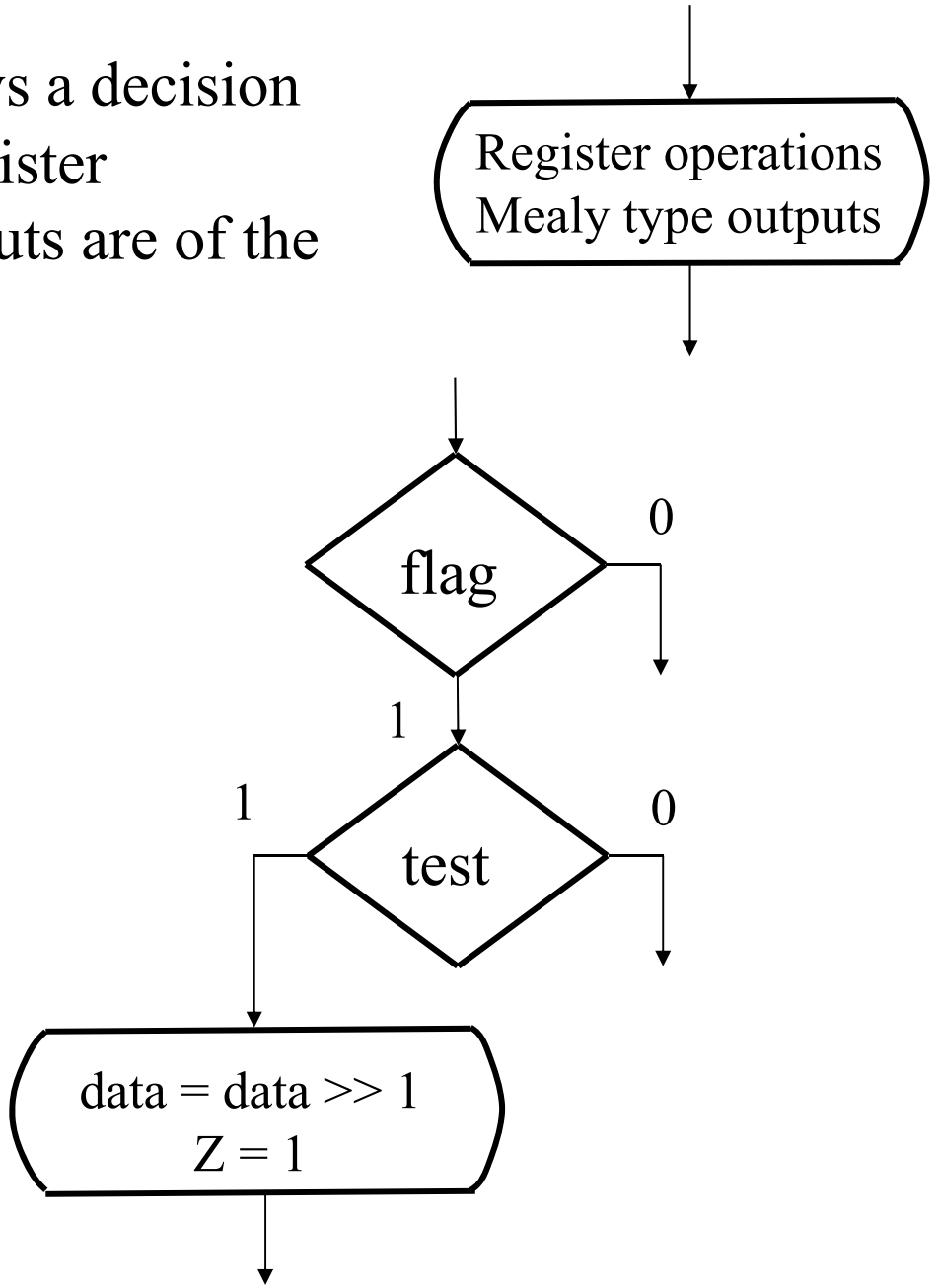


Decision box has one entry point but may have multiple exit points dependant on the decisions made.



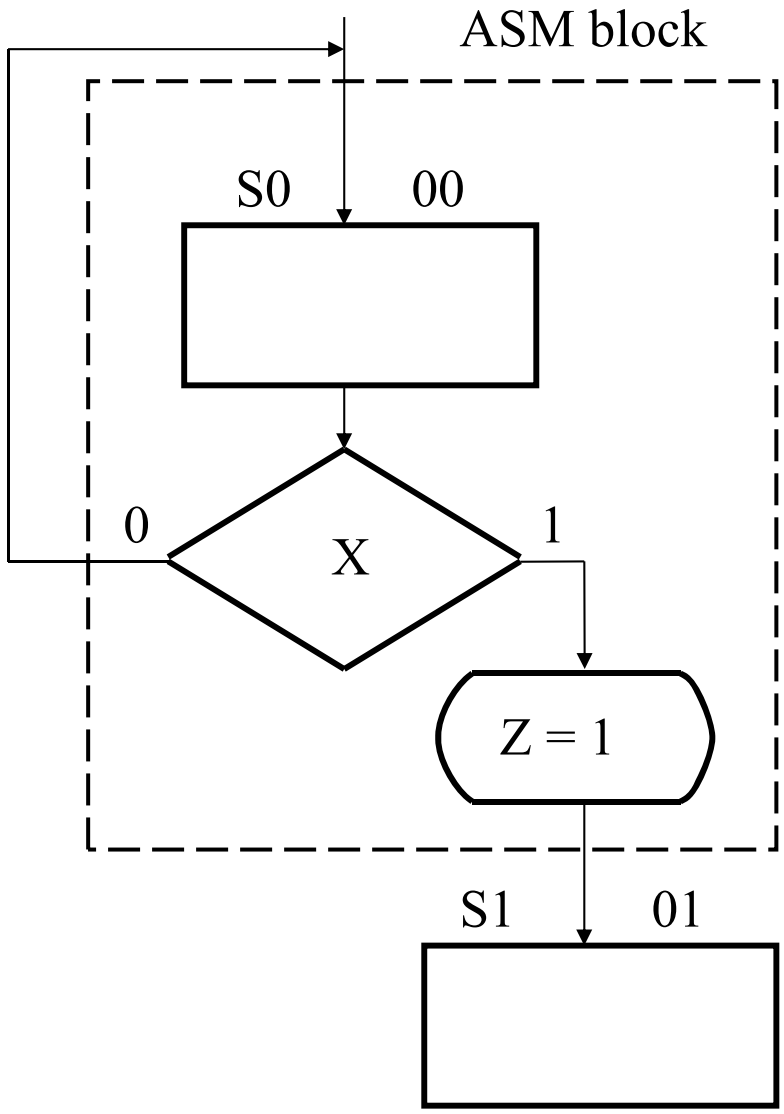
Conditional box: This follows a decision box and specifies conditional register operations and outputs. The outputs are of the Mealy type.

- Status of flag is checked.
- If flag = 1 check status of test
- If test = 1 shift data right by one place.
- Assert output Z = 1.

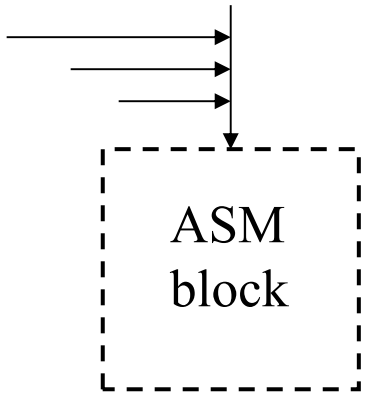


ASM block

An ASM block consists of a state box and all the decision boxes and conditional boxes connected to it. There is one entry point to the block. There may be more than one exit point. Each ASM block represents the state of the system during one clock cycle.

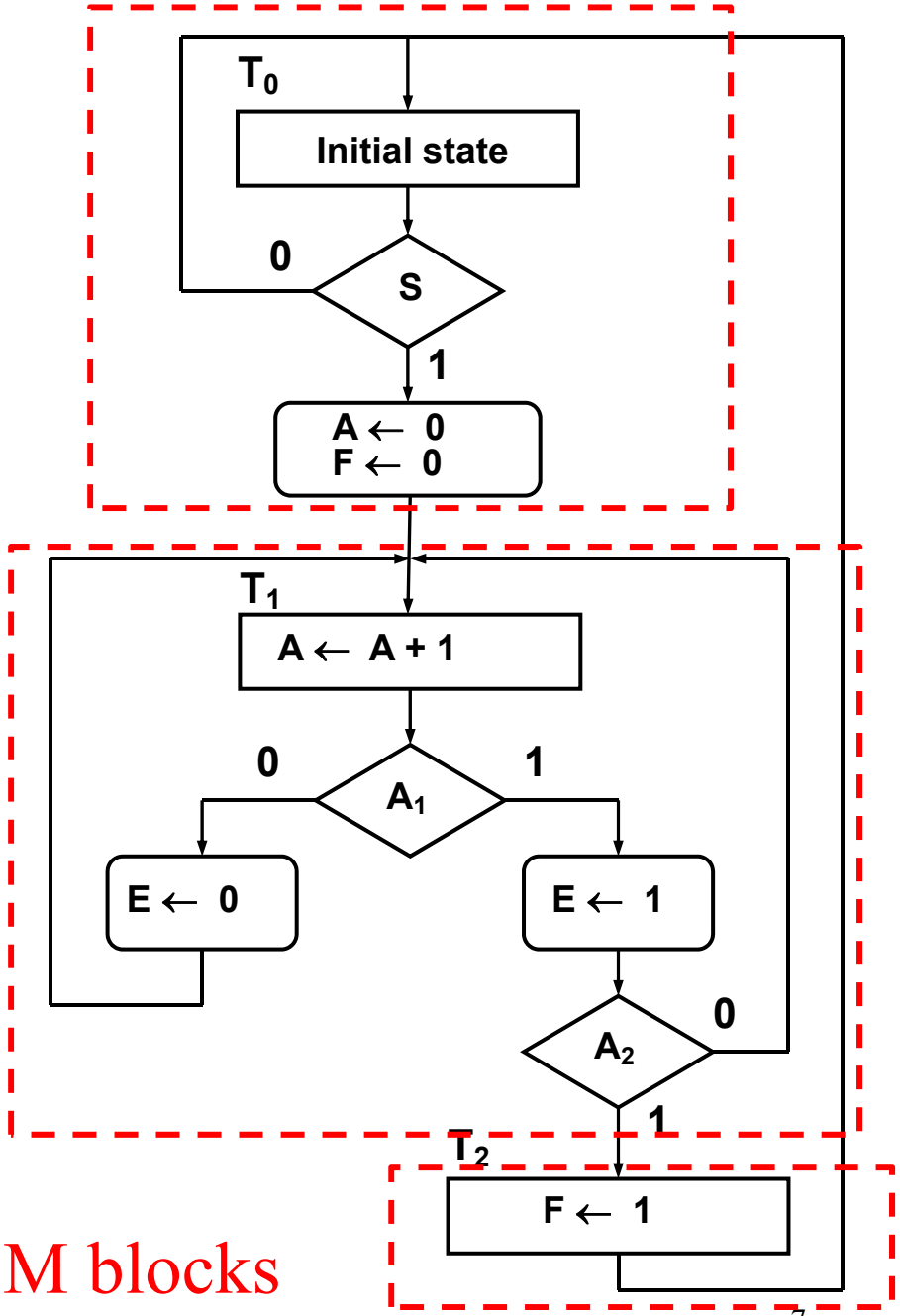


Two or more block exit paths may go to the same block entry point.



ASM Charts

- Abstraction of the functionality of a sequential machine.
- Organised into blocks
- One state box per block
- Optional number of decision boxes and conditional boxes per block
- State box represents the state of the machine between synchronizing clock edges
- Can represent both Moore and Mealy type finite state machines



3 ASM blocks

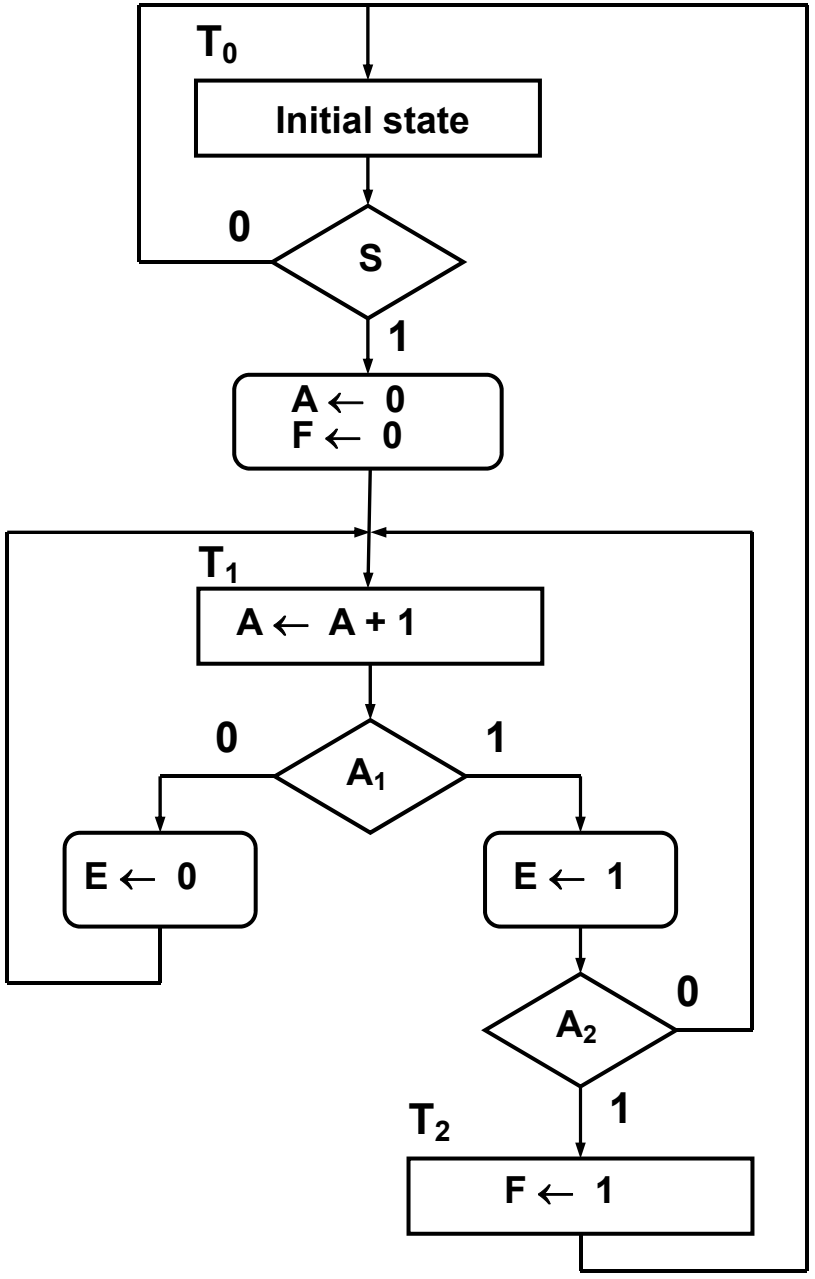
Register Operations

Data is stored in registers. Data transformations are described in the state or conditional boxes.

Consider A to be a 4-bit accumulator register:
$A = A_3A_2A_1A_0$

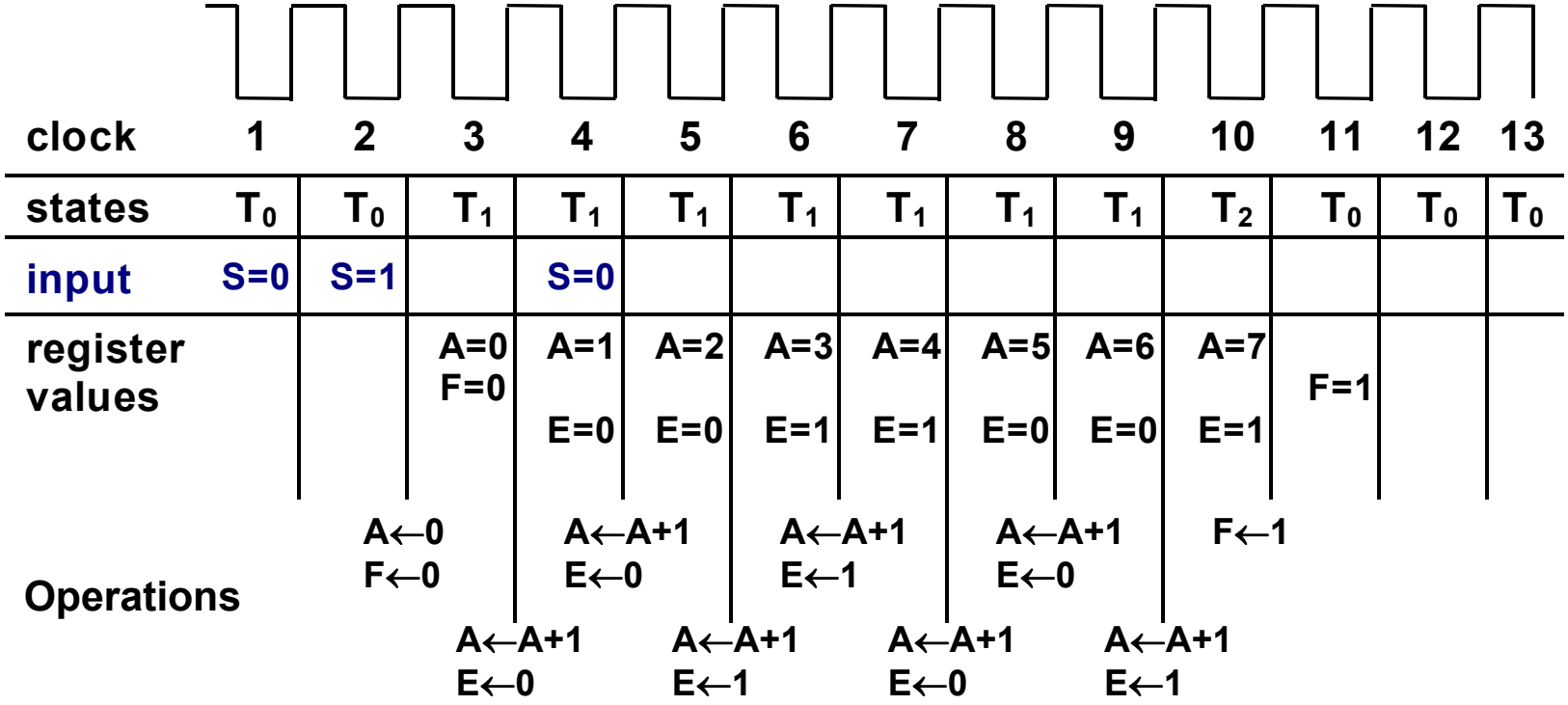
E and F are single-bit flip-flops.

$A \leftarrow B$	Transfer contents of register B into register A
$A \leftarrow 0$	Clear register A
$A \leftarrow A + 1$	Increment register A



Timing in ASM Charts

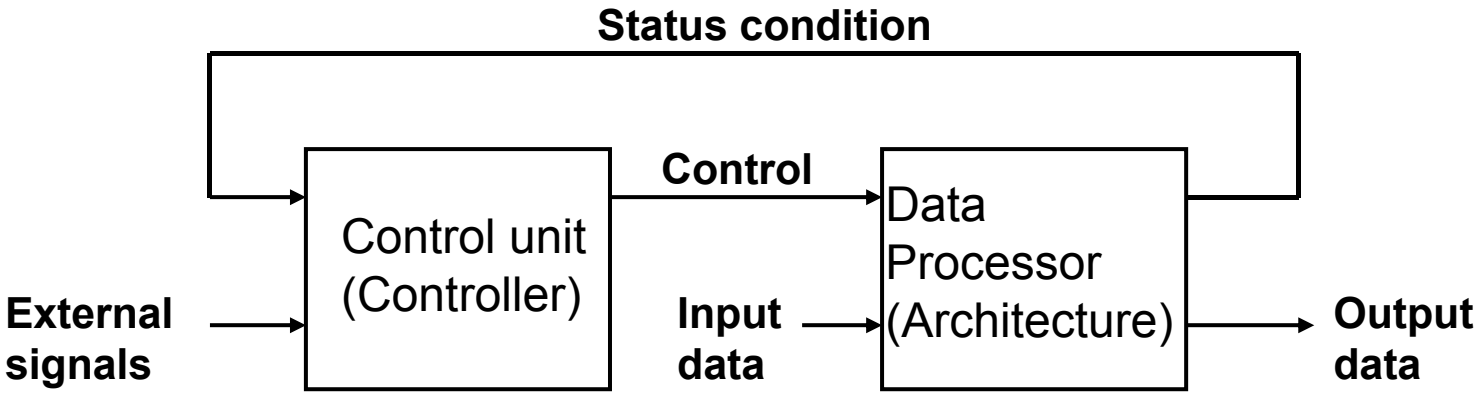
Operations in an ASM block occur in a *single clock cycle*.



$A = A_3A_2A_1A_0$

Digital Implementation

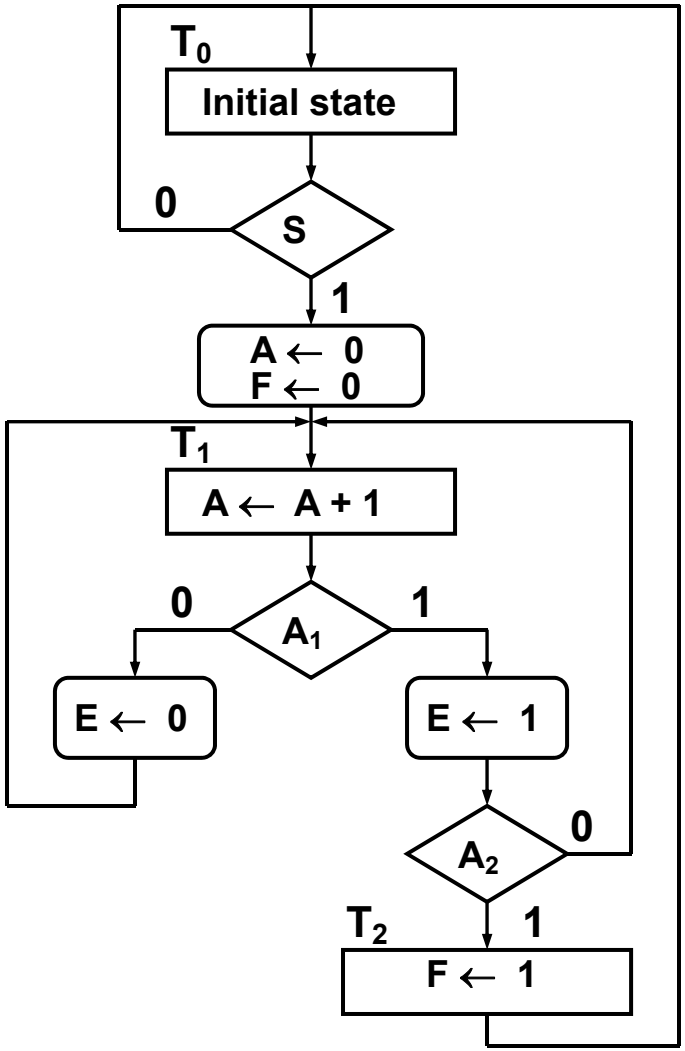
The ASM chart can be used to obtain the described digital system in the form of a Data Processor and Controller.

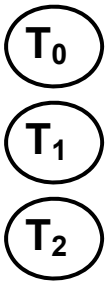


The controller logic can be obtained from the decision boxes and the state transitions.

The data processor can be obtained from the operations described in the state boxes and conditional boxes..

Controller





Assign codes to states:

$T_0 = 00$

$T_1 = 01$

$T_2 = 11$

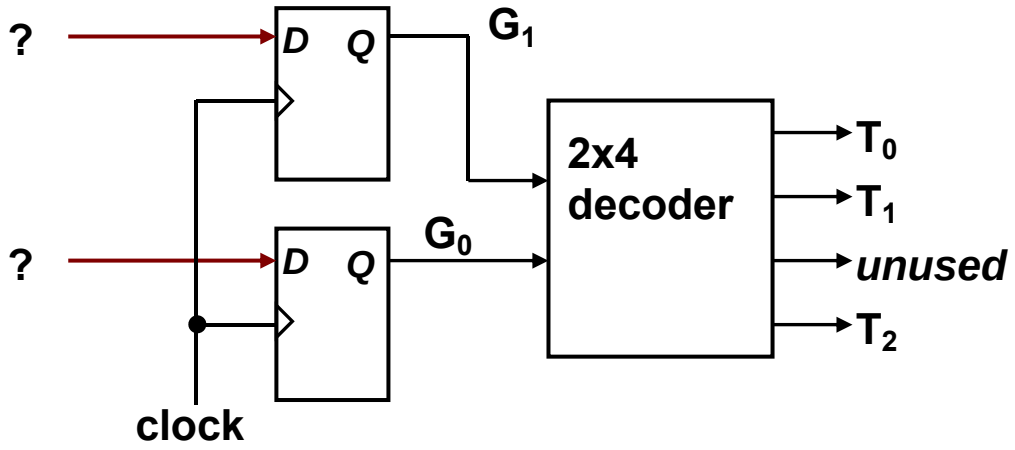
Present state		inputs			Next state		outputs		
G_1	G_0	S	A_1	A_2	G_1'	G_0'	T_0	T_1	T_2
0	0	0	X	X	0	0	1	0	0
0	0	1	X	X	0	1	1	0	0
0	1	X	0	X	0	1	0	1	0
0	1	X	1	0	0	1	0	1	0
0	1	X	1	1	1	1	0	1	0
1	1	X	X	X	0	0	0	0	1

Inputs from conditions in decision boxes.
Outputs = present state of controller.

- Register operations are:
  - Counter incremented ($A \leftarrow A + 1$) when state = T_1.
  - Counter cleared ($A \leftarrow 0$) when state = T_0 and S = 1.
  - E is set ($E \leftarrow 1$) when state = T_1 and $A_1 = 1$.
  - E is cleared ($E \leftarrow 0$) when state = T_1 and $A_1 = 0$.
  - F is set ($F \leftarrow 1$) when state = T_2.

- Datapath elements:
  - One 4-bit synchronous counter with clear/increment
  - Two D-type flip-flops needed for E and F

- The flip-flop input functions can be obtained directly from the state table by inspection.
- For D-type flip-flops, $Q_{next} = D$
- A decoder is then used to provide signals to drive datapath elements defined by the different states.

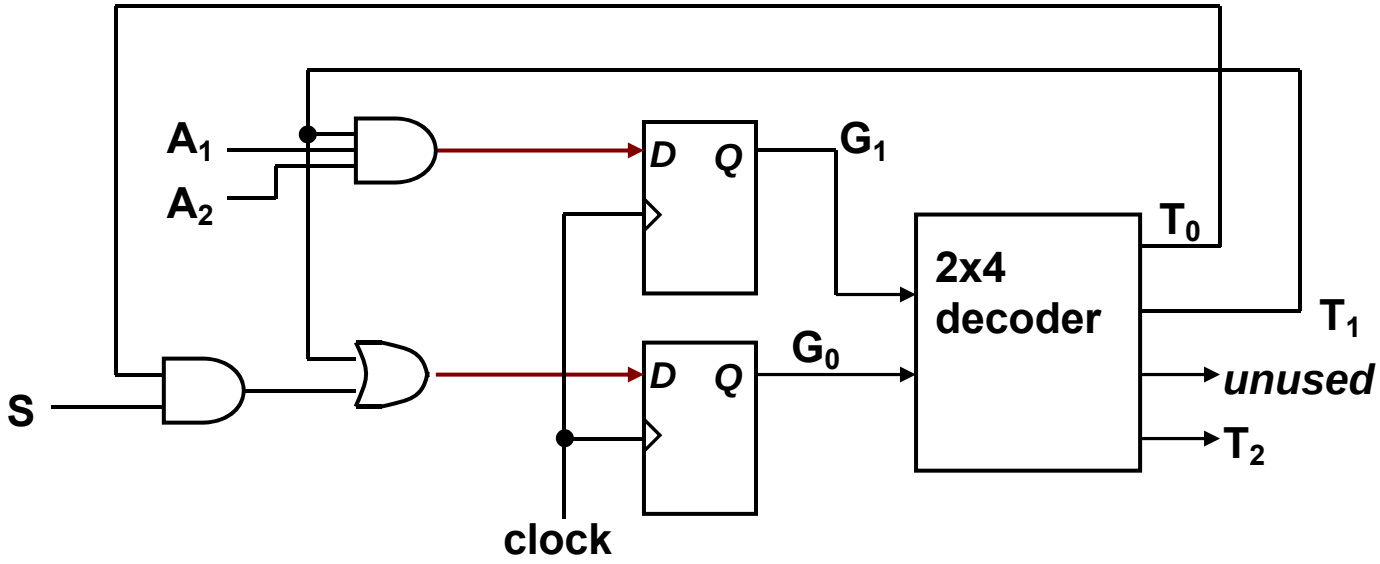


The inputs of the D flip-flops for G_1 and G_0 can be determined from the state table.

Present state		inputs			Next state		outputs		
G_1	G_0	S	A_1	A_2	G_1'	G_0'	T_0	T_1	T_2
0	0	0	X	X	0	0	1	0	0
0	0	1	X	X	0	1	1	0	0
0	1	X	0	X	0	1	0	1	0
0	1	X	1	0	0	1	0	1	0
0	1	X	1	1	1	1	0	1	0
1	1	X	X	X	0	0	0	0	1

$$G_1 = T1.A1.A2$$
$$G_0 = T0.S + T1$$

$$G_1 = T_1.A_1.A_2$$
$$G_0 = T_0.S + T_1$$



ROM based implementation

It is possible to use a ROM to replace the decoder and all of the gates in the input gates to the flip-flops.