BLG222E – Computer Organization

Gökhan Ince, PhD

gokhan.ince@itu.edu.tr

Office: EEB 4310

Tentative course outline

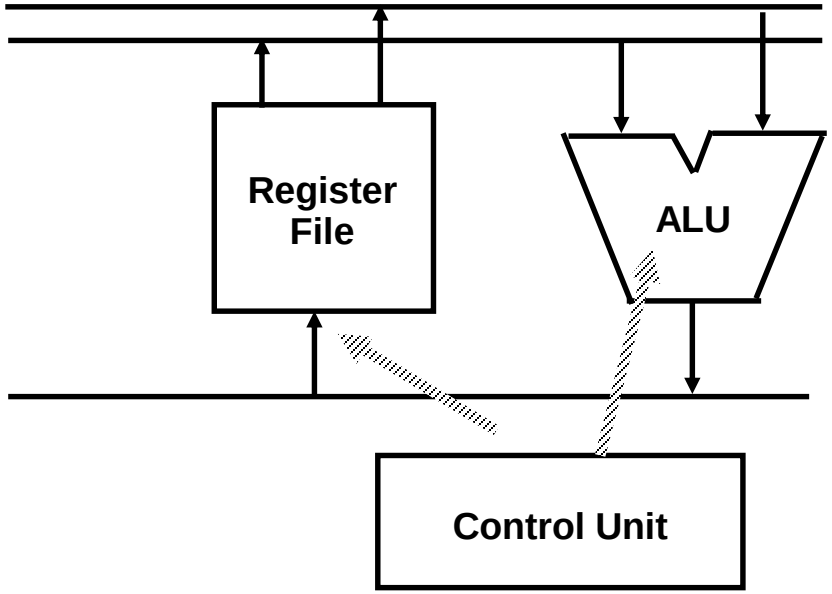
1) Introduction, combinational circuits, decoders, multiplexers **(1.1-2.3)**
2) Registers, ripple counters, memory units **(2.4, 2.5, 2.6, 2.7)**
3) Register transfer language (RTL), Bus, memory transfers **(4.1, 4.2, 4.3)**
4) Arithmetic operations, logical operations, shift operations, ALU **(4.4, 4.5, 4.6, 4.7)**
5) Instruction codes, timing and control, instruction cycles **(5.1-5.5)**
6) Memory-reference instructions, IO instructions **(5.6, 5.7)**
7) Design of a complete computer **(5.8, 5.9, 5.10)**
8) Midterm 1
9) Micro-programmed control, address sequencing **(7.1, 7.2)**
10) Design of control unit **(7.3,7.4)**
11) Stack organization, reverse polish notation **(8.3)**
12) National Holiday
13) Midterm 2
14) RISC/CISC processors

CENTRAL PROCESSING UNIT

- Introduction
- General Register Organization
- Stack Organization

MAJOR COMPONENTS OF CPU

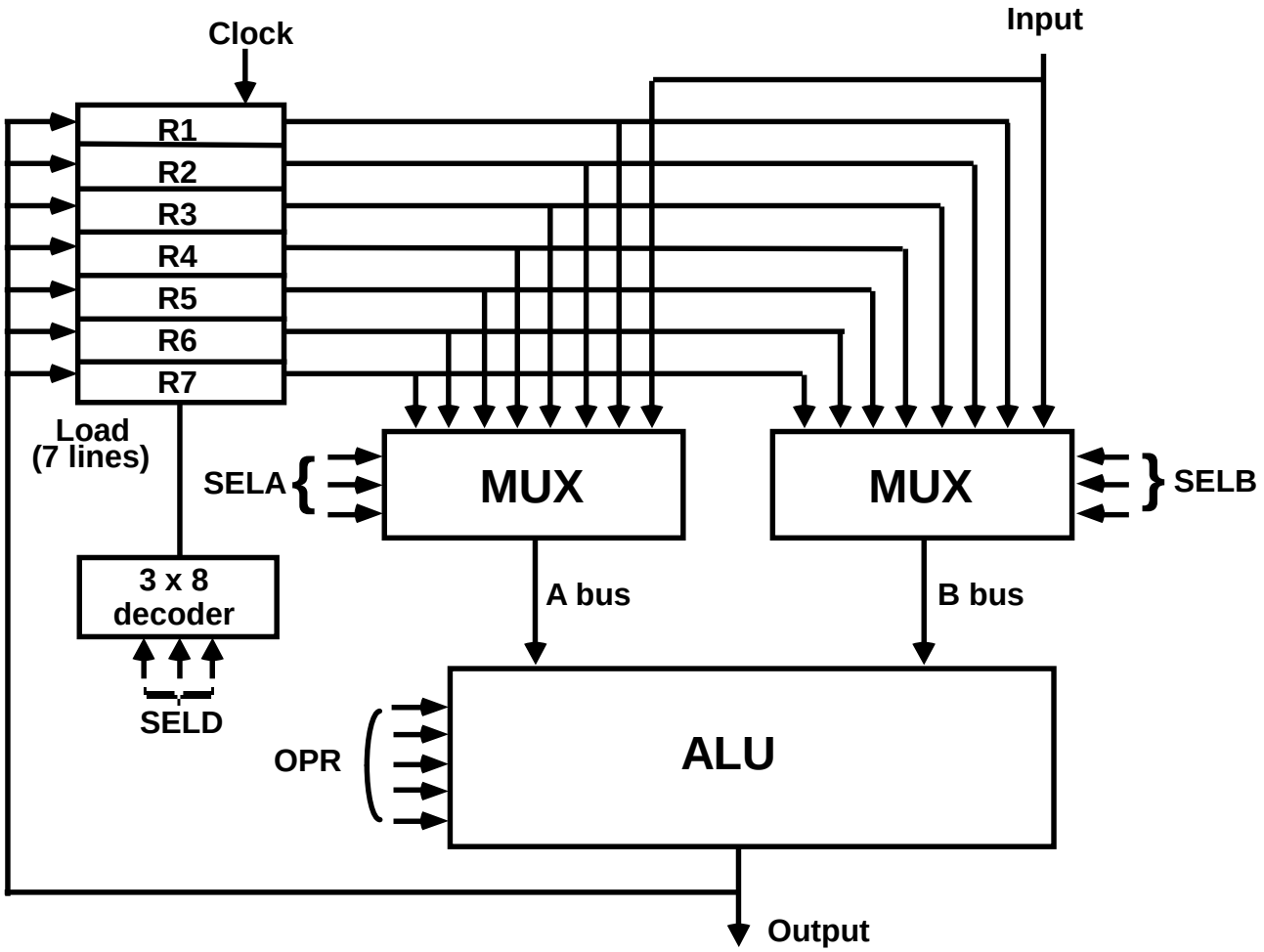
- **Storage Components**
  - Registers
  - Flags
- **Execution (Processing) Components**
  - Arithmetic Logic Unit (ALU)
    - Arithmetic calculations, Logical computations, Shifts/Rotates
- **Transfer Components**
  - Bus
- **Control Components**
  - Control Unit



REGISTERS

- In Basic Computer, there is only one general purpose register, the Accumulator (AC)
- In modern CPUs, there are many general purpose registers
- It is advantageous to have many registers
  - Transfer between registers within the processor are relatively fast
  - Going “off the processor” to access memory is much slower

GENERAL REGISTER ORGANIZATION



OPERATION OF CONTROL UNIT

The control unit

Directs the information flow through ALU by

- Selecting various *Components* in the system
- Selecting the *Function* of ALU

Example: R1 ← R2 + R3

[1] MUX A selector (SELA): BUS A ← R2

[2] MUX B selector (SELB): BUS B ← R3

[3] ALU operation selector (OPR): ALU to ADD

[4] Decoder destination selector (SELD): R1 ← Out Bus

Control Word

Encoding of register selection fields

Binary Code	SELA	SELB	SELD
000	Input	Input	None
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

ALU CONTROL

Encoding of ALU operations

OPR Select	Operation	Symbol
00000	Transfer A	TSFA
00001	Increment A	INCA
00010	ADD A + B	ADD
00101	Subtract A - B	SUB
00110	Decrement A	DECA
01000	AND A and B	AND
01010	OR A and B	OR
01100	XOR A and B	XOR
01110	Complement A	COMA
10000	Shift right A	SHRA
11000	Shift left A	SHLA

Examples of ALU Microoperations

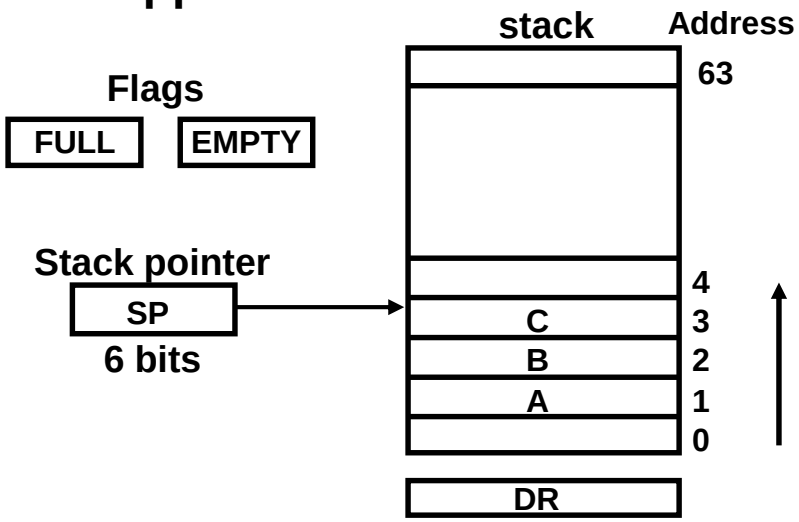
	Symbolic Designation				
Microoperation	SELA	SELB	SELD	OPR	Control Word
R1 ← R2 – R3	R2	R3	R1	SUB	010 011 001 00101
R4 ← R4 ∨ R5	R4	R5	R4	OR	100 101 100 01010
R6 ← R6 + 1	R6	-	R6	INCA	110 000 110 00001
R7 ← R1	R1	-	R7	TSFA	001 000 111 00000
Output ← R2	R2	-	None	TSFA	010 000 000 00000
Output ← Input	Input	-	None	TSFA	000 000 000 00000
R4 ← shl R4	R4	-	R4	SHLA	100 000 100 11000
R5 ← 0	R5	R5	R5	XOR	101 101 101 01100

REGISTER STACK ORGANIZATION

Stack

- Very useful feature for nested subroutines, nested interrupt services
- Also efficient for arithmetic expression evaluation
- Storage which can be accessed in LIFO
- Pointer: SP
- Only PUSH and POP operations are applicable

Register Stack



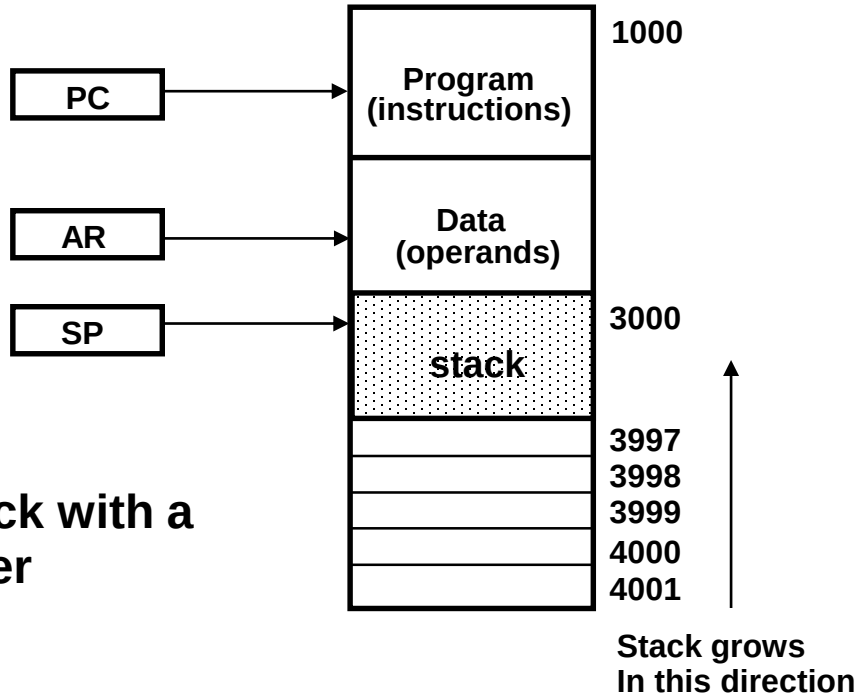
Push, Pop operations

/* Initially, SP = 0, EMPTY = 1, FULL = 0 */

PUSH	POP
SP ← SP + 1	DR ← M[SP]
M[SP] ← DR	SP ← SP – 1
If (SP = 0) then (FULL ← 1)	If (SP = 0) then (EMPTY ← 1)
EMPTY ← 0	FULL ← 0

MEMORY STACK ORGANIZATION

Memory with Program, Data, and Stack Segments



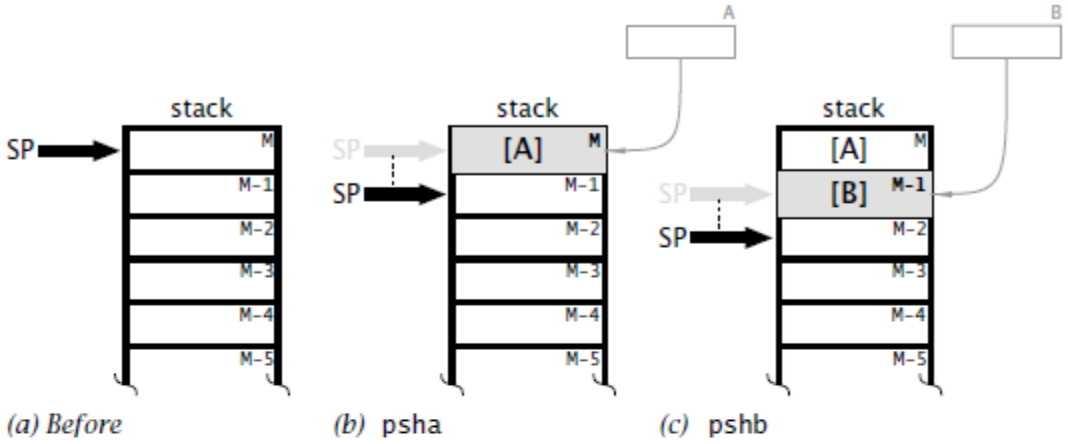
- A portion of memory is used as a stack with a processor register as a stack pointer

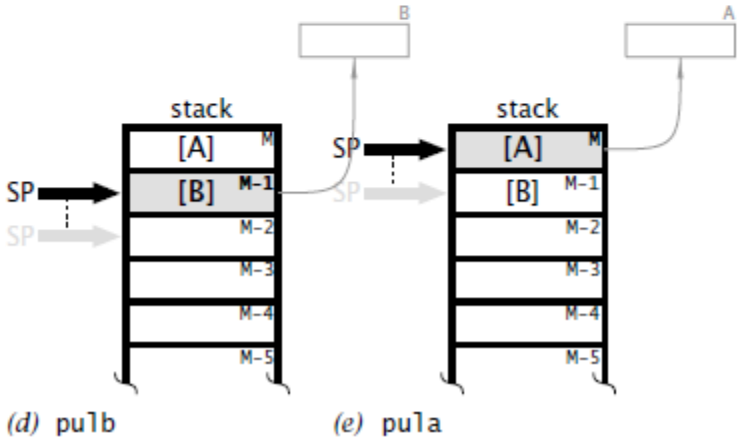
- PUSH: $SP \leftarrow SP - 1$
  $M[SP] \leftarrow DR$
- POP: $DR \leftarrow M[SP]$
  $SP \leftarrow SP + 1$

- Most computers do not provide hardware to check stack overflow (full stack) or underflow (empty stack) → must be done in software
- The advantage of memory stack is that the CPU can refer to it without having to specify an address.

Memory Stack Organization

- A stack may also be constructed so that SP **points** to the next empty location





- PUSH: $M[SP] \leftarrow A$
$SP \leftarrow SP - 1$

- POP: $SP \leftarrow SP + 1$
$DR \leftarrow M[SP]$

REVERSE POLISH NOTATION

- **Arithmetic Expressions: A + B**

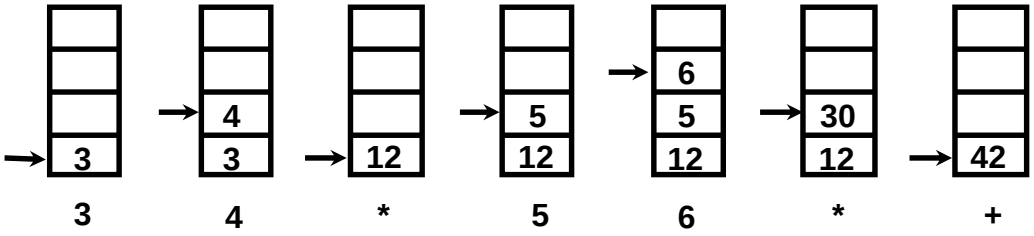
  A + B Infix notation
  + A B Prefix or Polish notation
  A B + Postfix or reverse Polish notation

  - The reverse Polish notation is very suitable for stack manipulation

- **Evaluation of Arithmetic Expressions**

  Any arithmetic expression can be expressed in parenthesis-free Polish notation, including reverse Polish notation

(3 * 4) + (5 * 6) ⇒ 3 4 * 5 6 * +

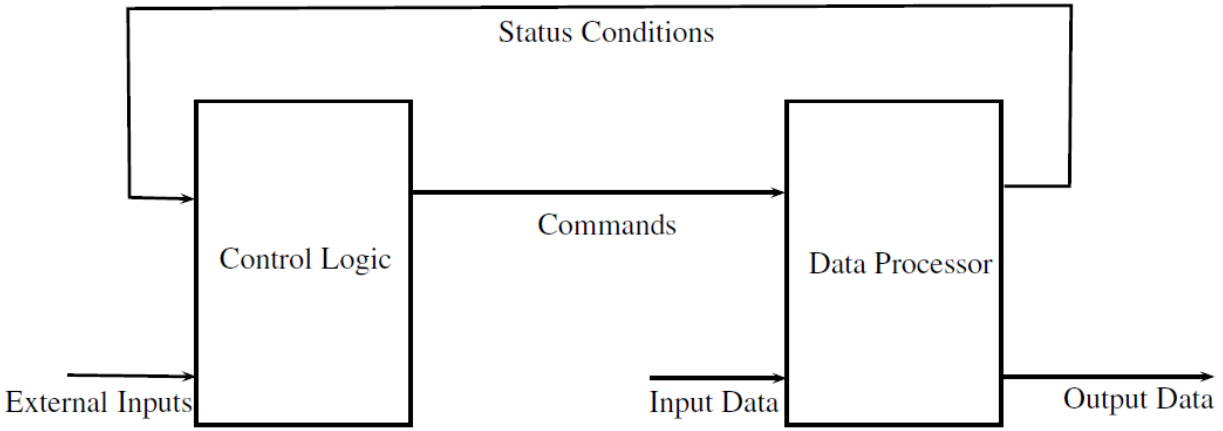


PROCESSOR ORGANIZATION

- In general, most processors are organized in one of 3 ways
  - Single register (Accumulator) organization
    - » Basic Computer is a good example
    - » Accumulator is the only general purpose register
  - General register organization
    - » Used by most modern computer processors
    - » Any of the registers can be used as the source or destination for computer operations
  - Stack organization
    - » All operations are done using the hardware stack
    - » For example, an OR instruction will pop the two top elements from the stack, do a logical OR on them, and push the result on the stack

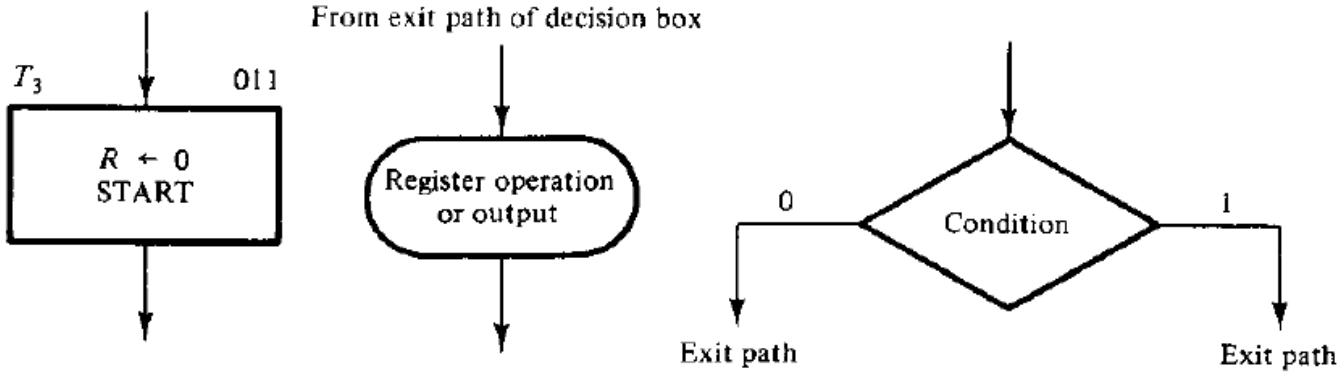
Digital System Design

- When designing digital hardware, a general approach is to handle data processing and control operations separately.
- The control logic and data processing tasks of a digital system are specified by means of a hardware algorithm.
- One of the common ways in representing an algorithm is by using a flowchart.

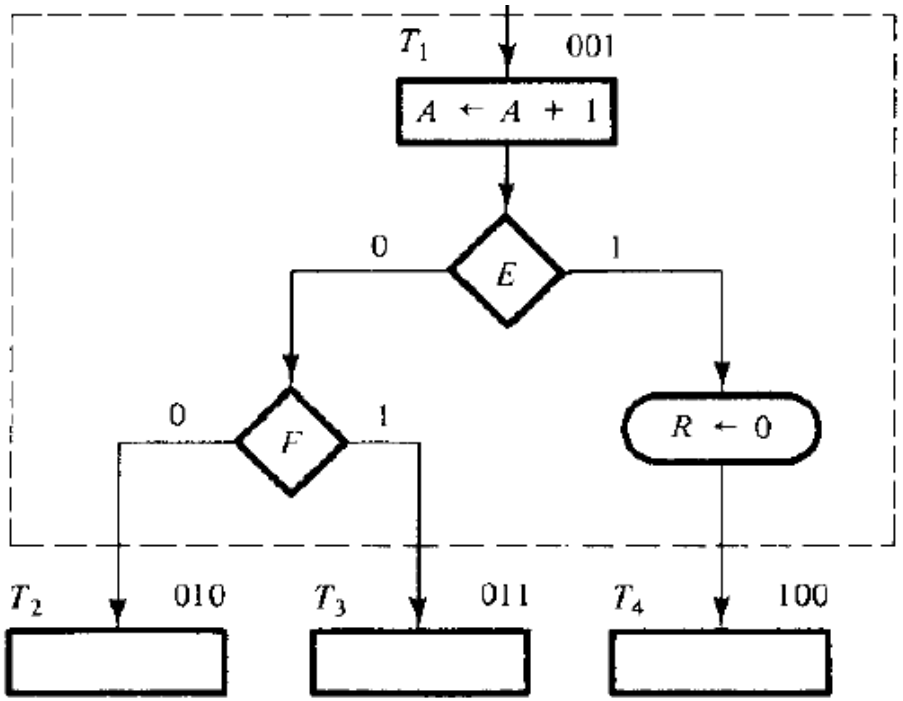


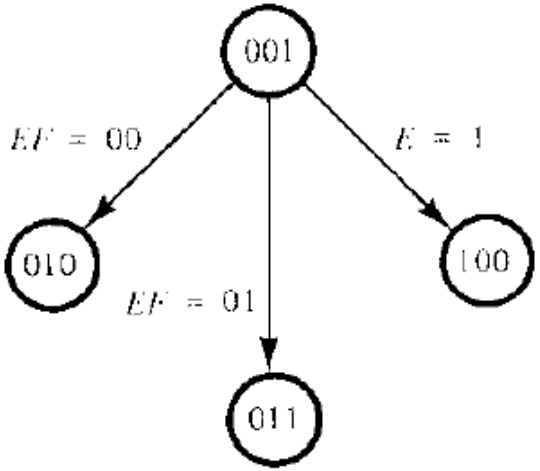
Algorithmic State Machines

- Algorithmic state machine(ASM) is a special type of flowchart that has been developed specifically to define digital hardware algorithms.
- ASMs are composed of three basic elements:
  - The state box: Denoted as a rectangle within which are written register operations or output signals.
  - The decision box: Diamond shaped box describing the effect of an input on the control subsystem.
  - The conditional box: Oval shaped box unique to ASMs. Register operations or outputs listed in the box are generated during a given state provided that the input condition is satisfied.



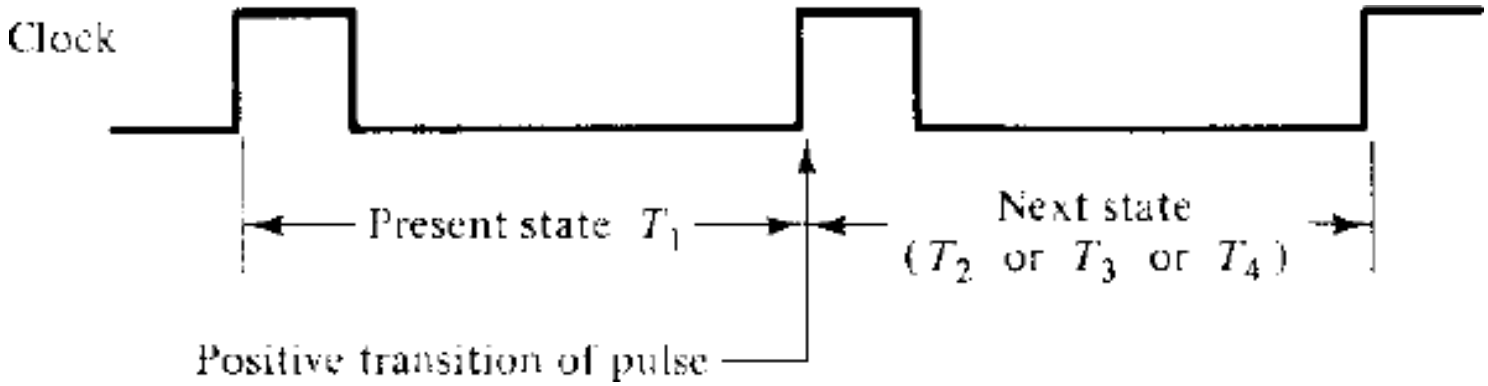
An ASM and an equivalent state chart





Timing properties of ASMs

- The major difference between an ASM and a flowchart is in interpreting its timing properties. In a flowchart every transition between elements is done sequentially while in an ASM operations in a block is done simultaneously. By the second rise of the clock signal's positive edge:
  - Register A is incremented
  - If E=1 register R is cleared
  - Depending on the values of E and F control is transferred to next state.

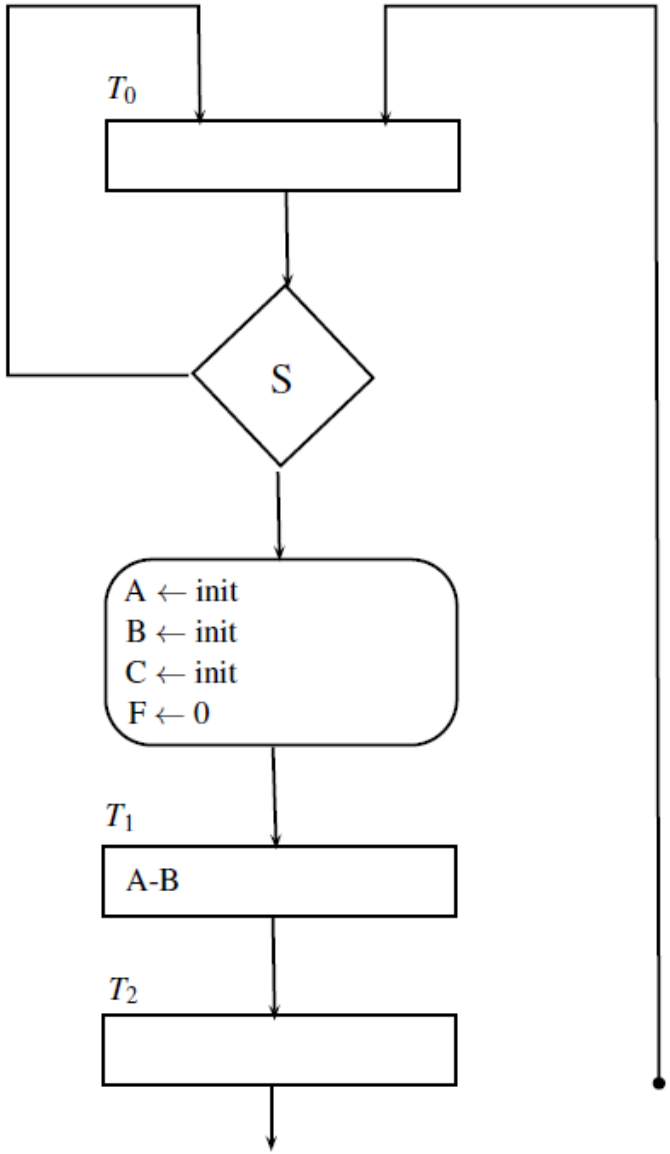


Example

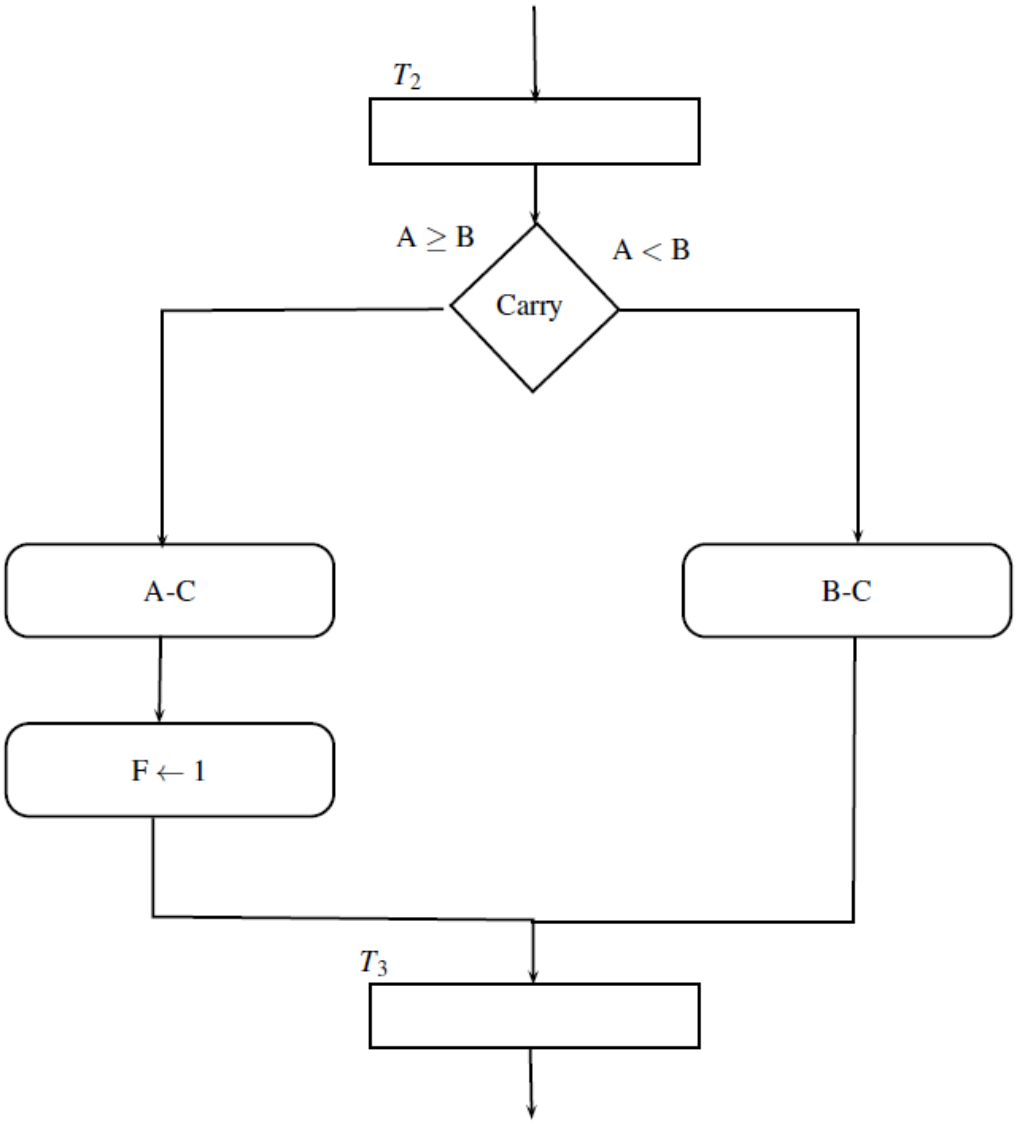
An ASM that calculates the largest of three positive integers is to be designed. Machine will kick off by an input signal switching to “1” and load the numbers successively to eight bit registers A, B and C. After the machine finishes its run and gets back to the starting state register “D” is going to hold the largest number D = Max(A,B,C). A combinational subtraction circuit is going to be used for comparison operations. Subtraction result is going to be held in a separate “carry” flag.

- Sketch the ASM diagram of the machine, defined above.
- Design and sketch the controller unit assigning each state to a D flip-flop.
- Design and sketch data unit. State the input signals of the units you have used.

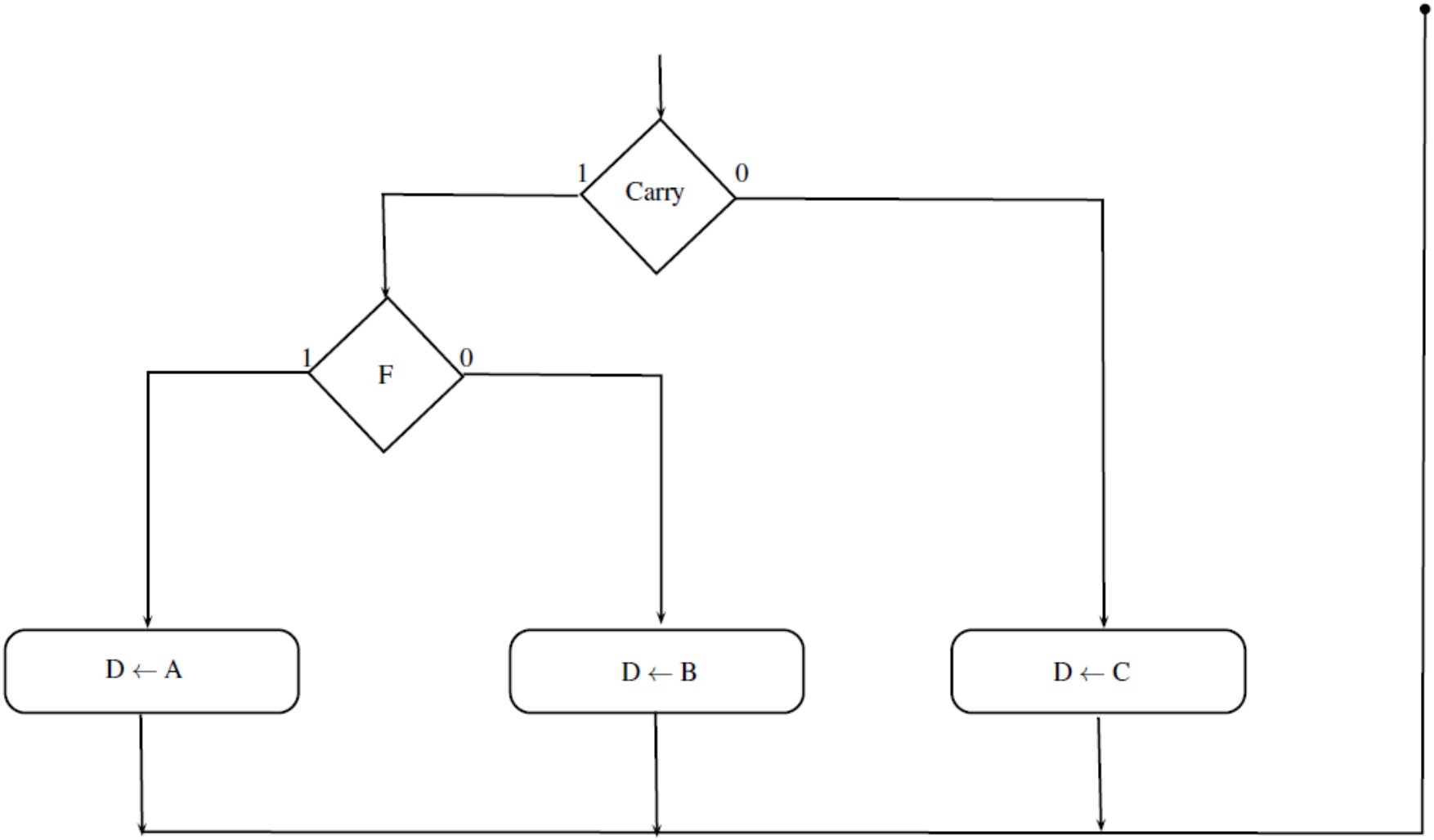
Example - ASM1



Example - ASM2

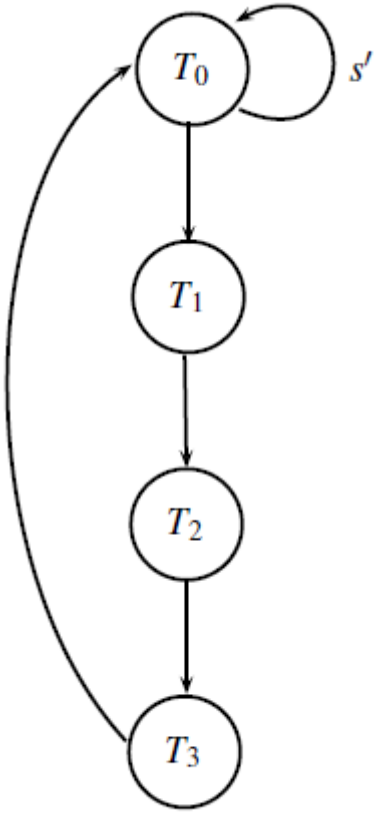


Example - ASM3



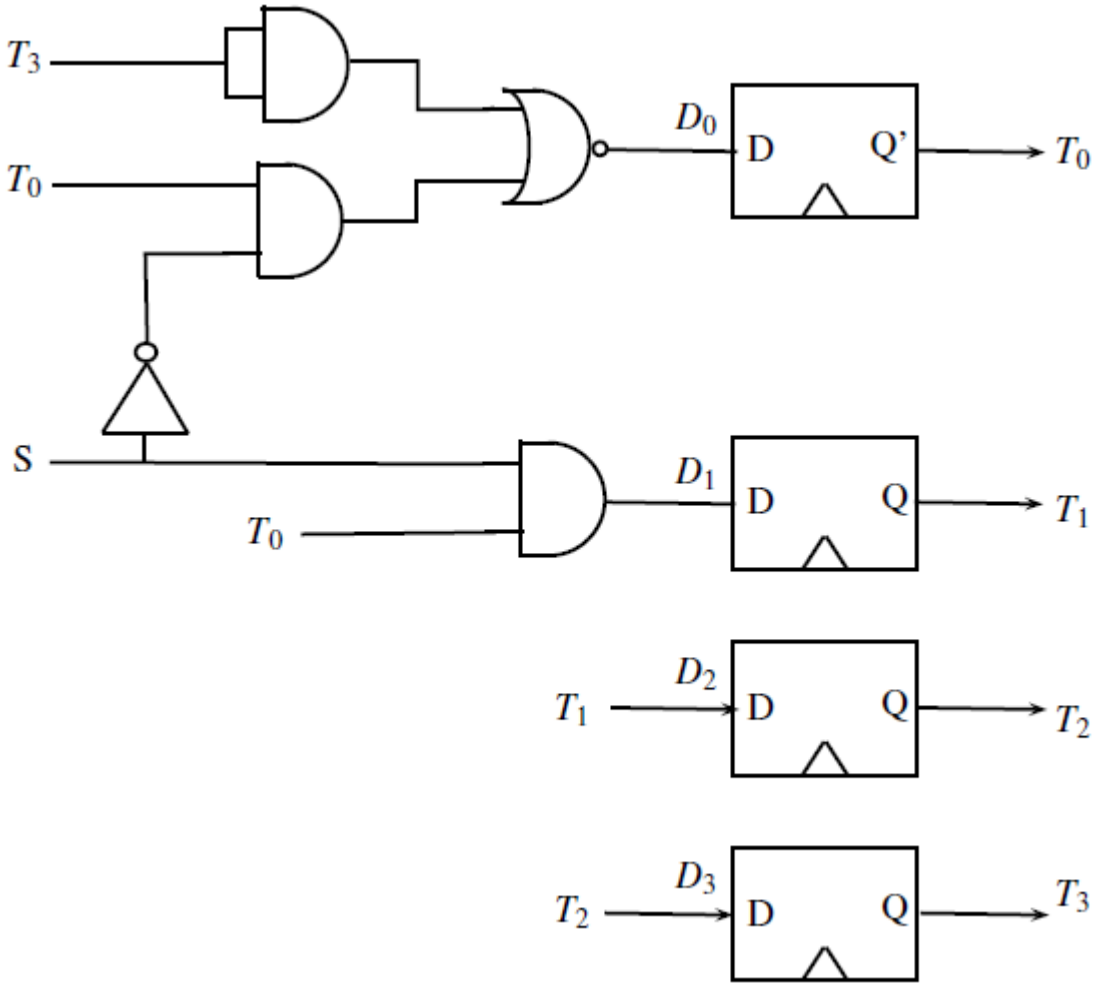
Example - States

- $T_0 = T_3 + s' T_0$
- $T_1 = s T_0$
- $T_2 = T_1$
- $T_3 = T_2$



Example - Control Unit

- $T_0 = T_3 + s'T_0$
- $T_1 = sT_0$
- $T_2 = T_1$
- $T_3 = T_2$



Example - Data Unit

- $L_A = L_B = L_C = sT_0$
- $L_D = T_3$
- $L_E = T_1 + T_2$
- $J_F = ET_2$
- $K_F = sT_0$
- $K_1 = T_3FE$
- $K_2 = T_1 + T_2E$
- $K_3 = T_1 + T_3F'E$
- $K_4 = T_2E'$
- $K_5 = T_2 + T_3E'$