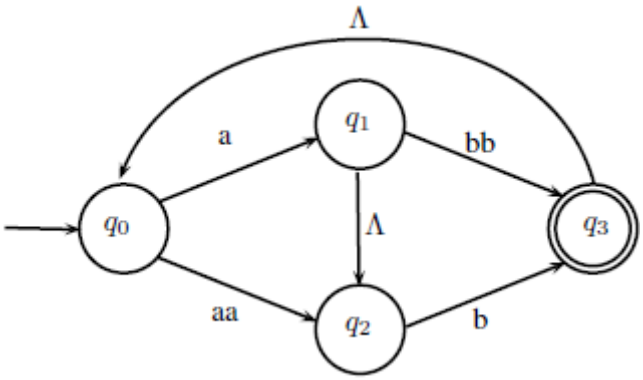
ITU Computer and Informatics Faculty
BLG311E Formal Languages and Automata 2013 Spring Term

24.05.2013

Duration: 120 minutes. Please deliver each question solved in a separate sheet.

FINAL

1. (40p) Considering the non-deterministic finite automaton(NFA) below



a) Heuristically derive the regular expression for the language recognized by this NFA.

b) Build the equivalent DFA for this NFA.

c) Produce the Type-3 grammar recognized by the DFA in (b). Give an equivalent Type-2 grammar.

Solution:

a) $L = (abb \vee ab \vee aab)^+$

b) First, we need to rearrange the NFA to make the length of each transition equal to 1.

$R(q_0) = \{q_0\}$

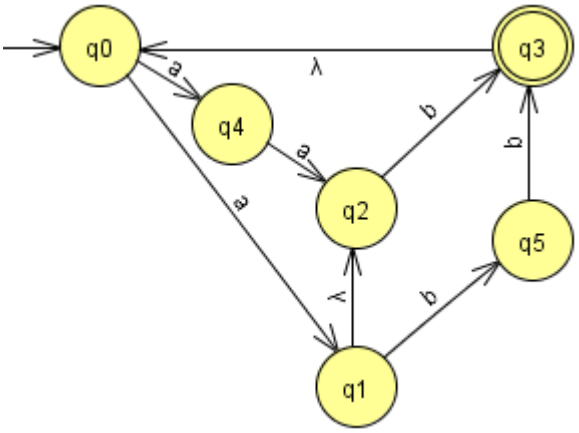
$R(q_1) = \{q_1, q_2\}$

$R(q_2) = \{q_2\}$

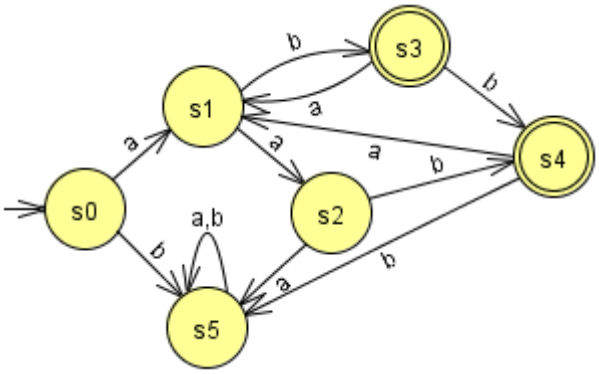
$R(q_3) = \{q_0, q_3\}$

$R(q_4) = \{q_4\}$

$R(q_5) = \{q_5\}$



$s_0 = R(q_0) = q_0$
$\delta(s_0, a) = \{R(q_1), R(q_4)\} = \{q_1, q_2, q_4\} \rightarrow s_1$
$\delta(s_0, b) = \emptyset$
$\delta(s_1, a) = R(q_2) = q_2 \rightarrow s_2$
$\delta(s_1, b) = \{R(q_3), R(q_5)\} = \{q_0, q_3, q_5\} \rightarrow \boldsymbol{s_3}$
$\delta(s_2, a) = \emptyset$
$\delta(s_2, b) = R(q_3) = \{q_0, q_3\} \rightarrow \boldsymbol{s_4}$
$\delta(s_3, a) = \{R(q_1), R(q_4)\} = \{q_1, q_2, q_4\} \rightarrow s_1$
$\delta(s_3, b) = R(q_3) = \{q_0, q_3\} \rightarrow s_4$
$\delta(s_4, a) = \{R(q_1), R(q_4)\} = \{q_1, q_2, q_4\} \rightarrow s_1$
$\delta(s_4, b) = \emptyset$
$\delta(\emptyset, a) = \emptyset \rightarrow s_5$



c) $< s_0 > ::= a < s_1 >$
$< s_1 > ::= a < s_2 > | b < s_3 > | b$
$< s_2 > ::= b < s_4 > | b$
$< s_3 > ::= a < s_1 > | b < s_4 > | b$
$< s_4 > ::= a < s_1 >$

$< s_0 >$ and $< s_4 >$ are the same. So production rule for $< s_4 >$ can be eliminated:

$< s_0 > ::= a < s_1 >$
$< s_1 > ::= a < s_2 > | b < s_3 > | b$
$< s_2 > ::= b < s_0 > | b$
$< s_3 > ::= a < s_1 > | b < s_0 > | b$

Type-2 equivalent: $S \rightarrow AS \mid A$
$A \rightarrow ab \mid abb \mid aab$

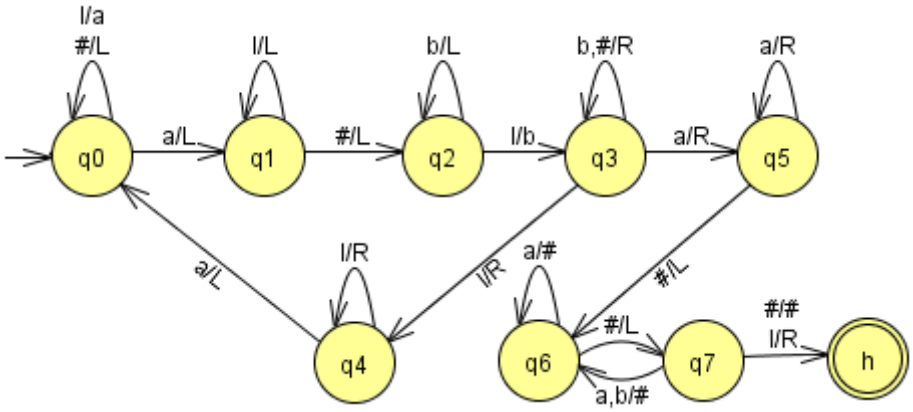
2. (30p) Design a Turing machine that can subtract two numbers a and b where $a \geq b > 0$. The numbers are placed on the tape as successive I symbols; where I count indicates the number. The initial configuration of the machine is the larger number is placed leftmost side of the tape with a preceding blank followed by a trailing blank and the smaller number. Initially r/w head is on the leftmost blank after the smaller number. After the machine halts, there will be I symbols placed leftmost side of the tape with a preceding blank; where I count indicates the difference. Some example computations can be given as:

- $\#IIIII\#III\underline{\#} \vdash_M{}^* \#II\underline{\#}$ $(5-3=2)$
- $\#III\#III\underline{\#} \vdash_M{}^* \#\#\underline{\#}$ $(3-3=0)$

Considering the information above give the state transition function of this Turing machine. You may use additional symbols such as γ in your function.

Solution:

q	σ	$\delta(q,\sigma)$
q_0	#	(q_0, L)
q_0	I	(q_0, a)
q_0	a	(q_1, L)
q_1	I	(q_1, L)
q_1	#	(q_2, L)
q_2	b	(q_2, L)
q_2	I	(q_3, b)
q_3	b	(q_3, R)
q_3	#	(q_3, R)
q_3	I	(q_4, R)
q_3	a	(q_5, R)
q_4	I	(q_4, R)
q_4	a	(q_0, L)
q_5	a	(q_5, R)
q_5	#	(q_6, L)
q_6	a	$(q_6, \#)$
q_6	#	(q_7, L)
q_7	a	$(q_6, \#)$
q_7	b	$(q_6, \#)$
q_7	I	(h, L)
q_7	#	$(h, \#)$



3. (40p) In a serial protocol, lets assume that every stream starts with a one byte character ":". After the reception of start character, one byte of address information and one byte of command word is received. If the address information matches with the address stored within an address register (namely reg. A), then the receiving unit writes the command byte into a command register (namely reg. C).

In this question you are asked to design a serial communication module(SCM) that works as follows:

1) SCM continously checks a buffer(COMBUF) if the one byte start character(":") is received.
2) When the start character is received SCM waits for the incoming address byte. If it matches A register, SCM yields to next step else resets to initial state.
3) SCM waits for the incoming command byte and writes it to C when data is received. Afterwards machine continues to initial state.
4) During the operations above, if a timeout occurs during the data communication(except starting character), machine resets to initial state.(See TCNT definition below)

To realize the machine defined above you are going to use the following three modules as blocks(Do not design the internal structure of them)

- Communication buffer(COMBUF): The reception is performed by a COMBUF module that has a logic 1 "data ready" and 8-bit data output. The data ready output is cleared to 0 with the start of a new byte.
- Equality comparator(EQCOMP): A combinatorial block that has two 8-bit data inputs and a logic 1 "equal" output.
- Timeout counter(TCNT): It has a logic 1 "count enable" signal input and a logic 1 "timeout" output set after a predefined period of time. The output is given for just one clock cycle and resets itself automatically.

When designing the (SCM), follow the steps given below

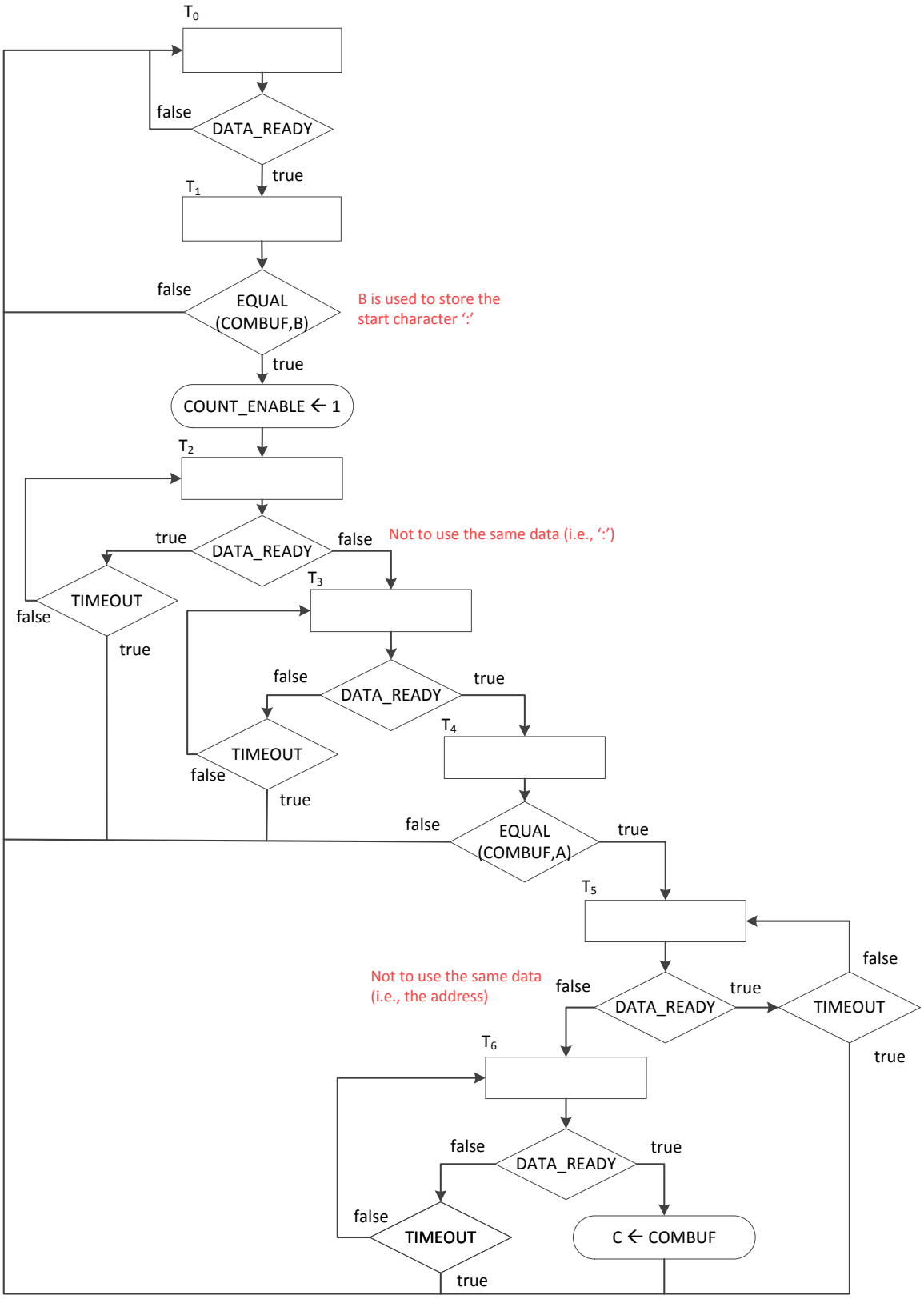
a) Draw the flowchart of an ASM that puts the command byte into register C when the address matches.

b) Implement the data unit by using COMBUF, EQCOMP and TCNT modules as well as registers A and C as building blocks.

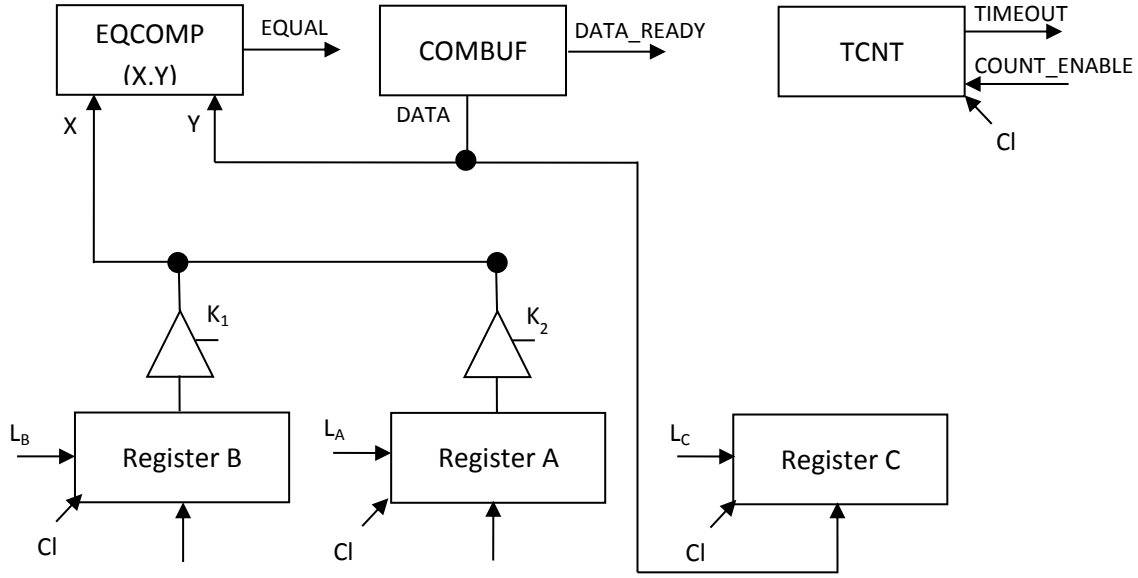
c) Design the control unit by encoding the state variables(try to use as small number of flip-flops as possible)

Solution:

a) Flowchart:



b) Data unit:



$$\boldsymbol{L_A = L_B = T_0,\ L_C = T_6 DATA\_READY \qquad K_1 = T_1,\ K_2 = T_4}$$
$$\boldsymbol{COUNT\_ENABLE = T_1 EQUAL}$$

c)Control unit:

$$T_0 = T_0\overline{DATA\_READY} + T_1\overline{EQUAL} + T_2 DATA\_READY\ TIMEOUT + T_3\overline{DATA\_READY}\ TIMEOUT + T_4\overline{EQUAL} + T_5 DATA\_READY\ TIMEOUT + T_6(\overline{DATA\_READY}\ TIMEOUT + DATA\_READY)$$
$$T_1 = T_0 DATA\_READY$$
$$T_2 = T_1 EQUAL + T_2 DATA\_READY\ \overline{TIMEOUT}$$
$$T_3 = T_2\overline{DATA\_READY} + T_3\overline{DATA\_READY}\ \overline{TIMEOUT}$$
$$T_4 = T_3 DATA\_READY$$
$$T_5 = T_4 EQUAL + T_5 DATA\_READY\ \overline{TIMEOUT}$$
$$T_6 = T_5\overline{DATA\_READY} + T_6\overline{DATA\_READY}\ \overline{TIMEOUT}$$

For state coding:

$T_0 \rightarrow \bar{M}_2\bar{M}_1\bar{M}_0$

$T_1 \rightarrow \bar{M}_2\bar{M}_1 M_0$

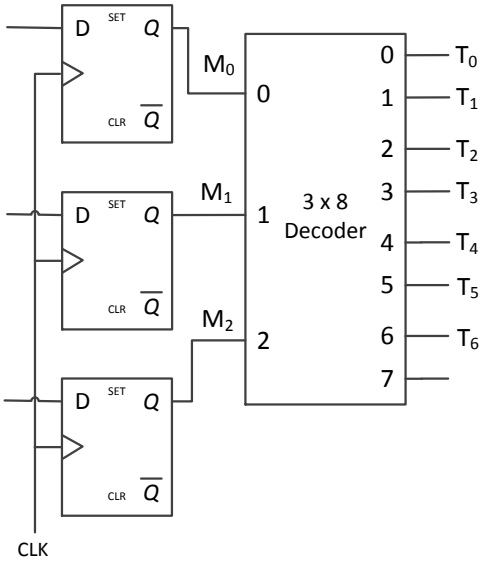
$T_2 \rightarrow \bar{M}_2 M_1\bar{M}_0$

$T_3 \rightarrow \bar{M}_2 M_1 M_0$

$T_4 \rightarrow M_2\bar{M}_1\bar{M}_0$

$T_5 \rightarrow M_2\bar{M}_1 M_0$

$T_6 \rightarrow M_2 M_1\bar{M}_0$



$$M_0 = T_1 + T_3 + T_5$$
$$= T_0 DATA\_READY + T_2\overline{DATA\_READY} + T_3\overline{DATA\_READY}\ \overline{TIMEOUT} + T_4 EQUAL + T_5 DATA\_READY\ \overline{TIMEOUT}$$

$$M_1 = T_2 + T_3 + T_6$$
$$= T_1 EQUAL + T_2 DATA\_READY\ \overline{TIMEOUT} + T_2 \overline{DATA\_READY} + T_3 \overline{DATA\_READY}\ \overline{TIMEOUT}$$
$$+ T_5 \overline{DATA\_READY} + T_6 \overline{DATA\_READY}\ \overline{TIMEOUT}$$
$$M_2 = T_4 + T_5 + T_6$$
$$= T_3 DATA\_READY + T_4 EQUAL + T_5 DATA\_READY \overline{TIMEOUT} + T_5 \overline{DATA\_READY}$$
$$+ T_6 \overline{DATA\_READY}\ \overline{TIMEOUT}$$