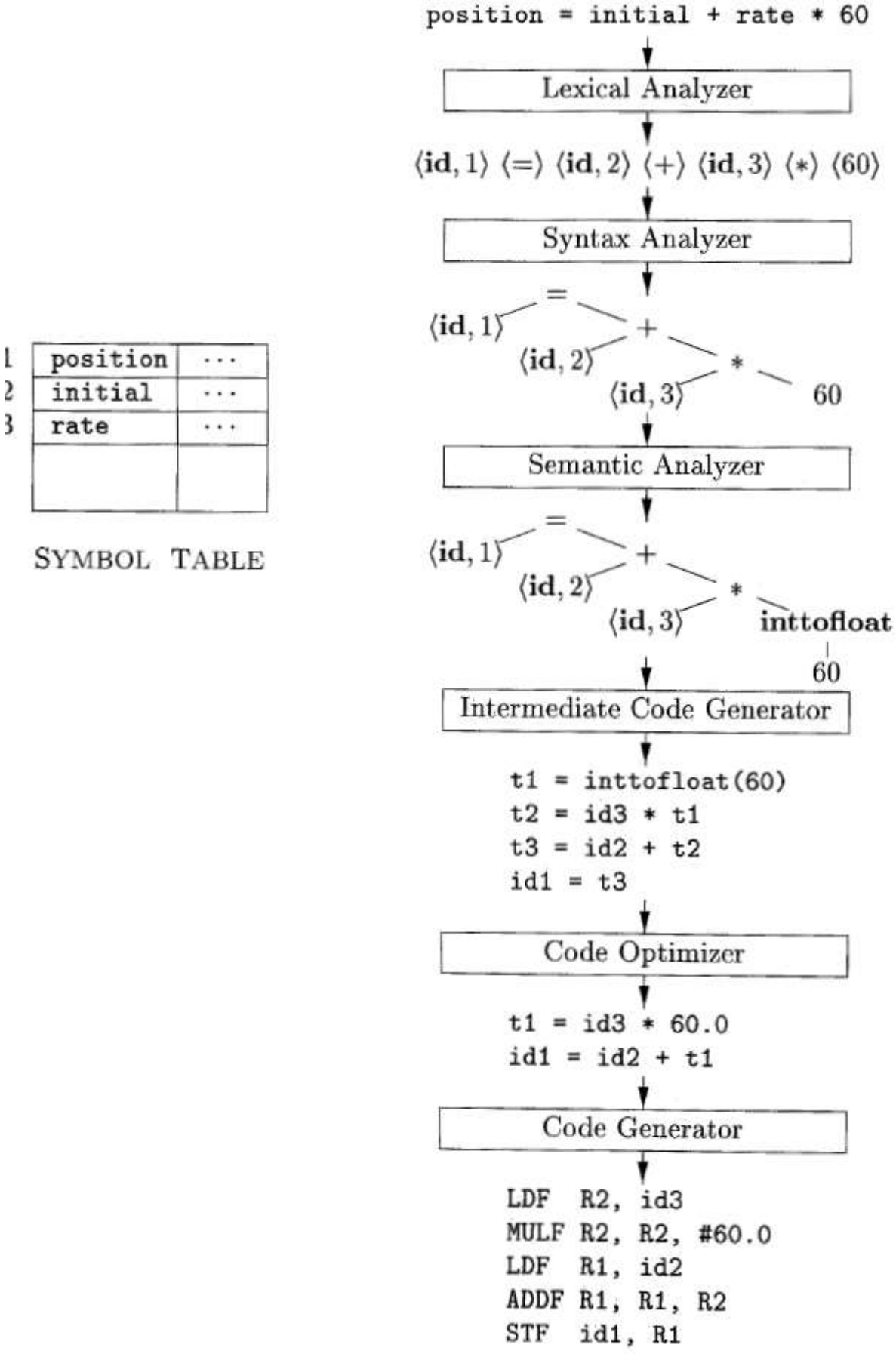
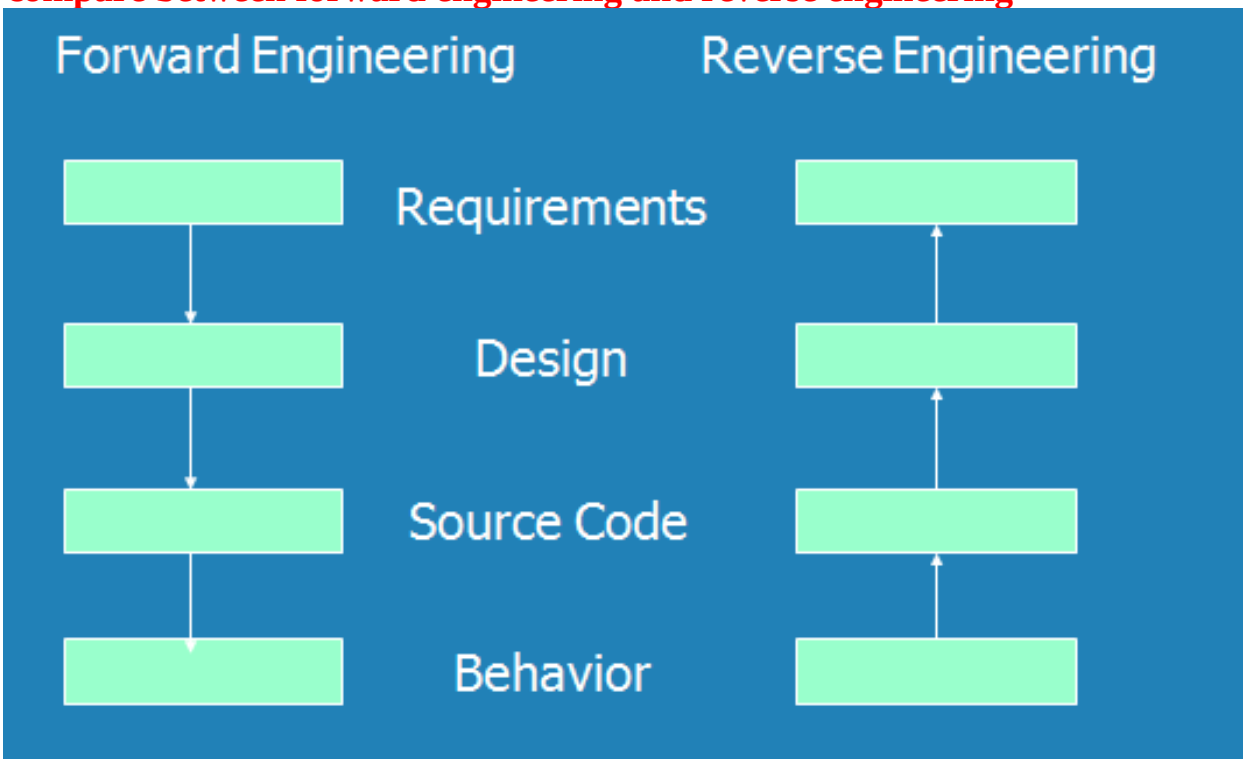
Revision – part 2

Lecture 5: Reverse Engineering

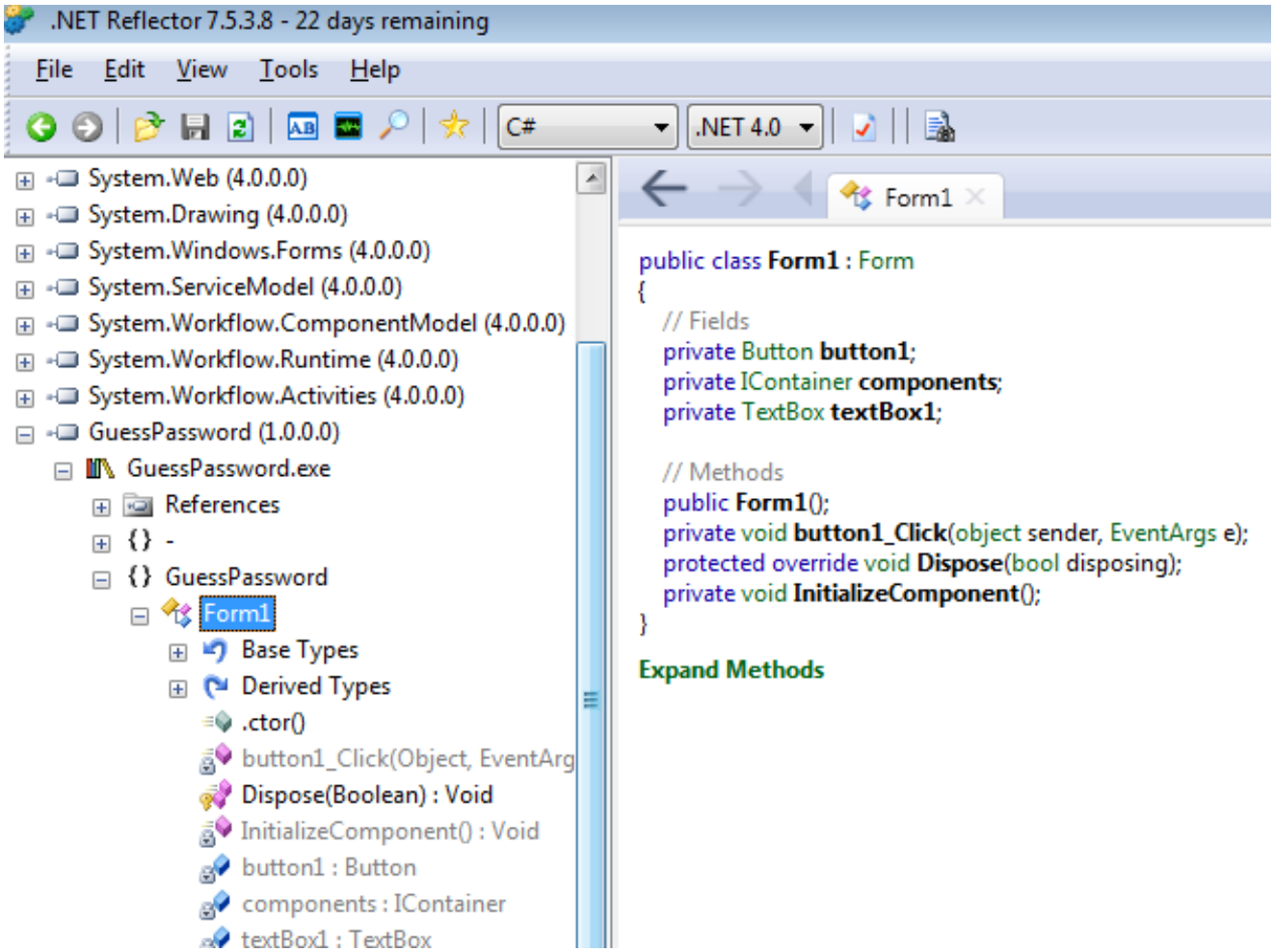
Q1 show the Phases of Compiler in converting the following statement into Assembly: position = initial + rate * 60 ;



Q2 compare between forward engineering and reverse engineering



سؤال : تعرف الـ reflector



Q3 WinRC-trans (Arabization tool) affects the (code – data- stack) segment?

Data

Lecture 6: Architecture review

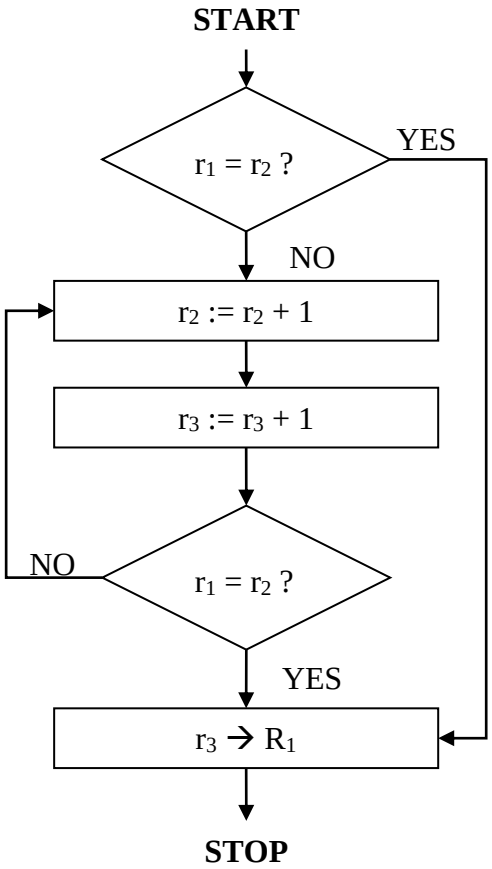
Q4 Consider the following program:

I_1: J(1, 2, 6)
I_2: S(2)
I_3: S(3)
I_4: J(1, 2, 6)
I_5: J(1, 1, 2)
I_6: T(3, 1)

i) Analyze the function of the program and Draw the flow diagram for the program
ii) Perform computation under this program with the initial configuration 8,4,2,0, …

Assume the initial configuration is : x,y,z. After k iterations, the value of first register is z+k.

Output is in first register. If x = y + k. then the output is z + x - y



Typical Configuration

x	y	z	

After k cycles round the loop

x	y+k	z+k	

If x = y + k :

z+k	y+k	z+k	

	R1	R2	R3	R4	R5		Next Instruction
Initial configuration	8	4	2	0	0	…	I_1
	8	4	2	0	0	…	I_2 (since $r_1 \neq r_2$)
	8	5	2	0	0	…	I_3
	8	5	3	0	0	…	I_4
	8	5	3	0	0	…	I_5 (since $r_1 \neq r_2$)
	8	5	3	0	0	…	I_2 (since $r_1 = r_1$)
	8	6	3	0	0	…	I_3
	8	6	4	0	0	…	I_4
	8	6	4	0	0	…	I_5 (since $r_1 \neq r_2$)
	8	6	4	0	0	…	I_2 (since $r_1 = r_1$)
	8	7	4	0	0	…	I_3
	8	7	5	0	0	…	I_4
	8	7	5	0	0	…	I_5 (since $r_1 \neq r_2$)
	8	7	5	0	0	…	I_2 (since $r_1 = r_1$)
	8	8	5	0	0	…	I_3
	8	8	6	0	0	…	I_4
	8	8	6	0	0	…	I_6 (since $r_1 = r_2$)
Final configuration	**6**	8	6	0	0	…	I_7 : STOP

Q5: Computer Organization vs Architecture

…. is related to Organization or Architecture?

Organization (=how) **differs** between different versions

Architecture (=specification) **is compatible** between different versions

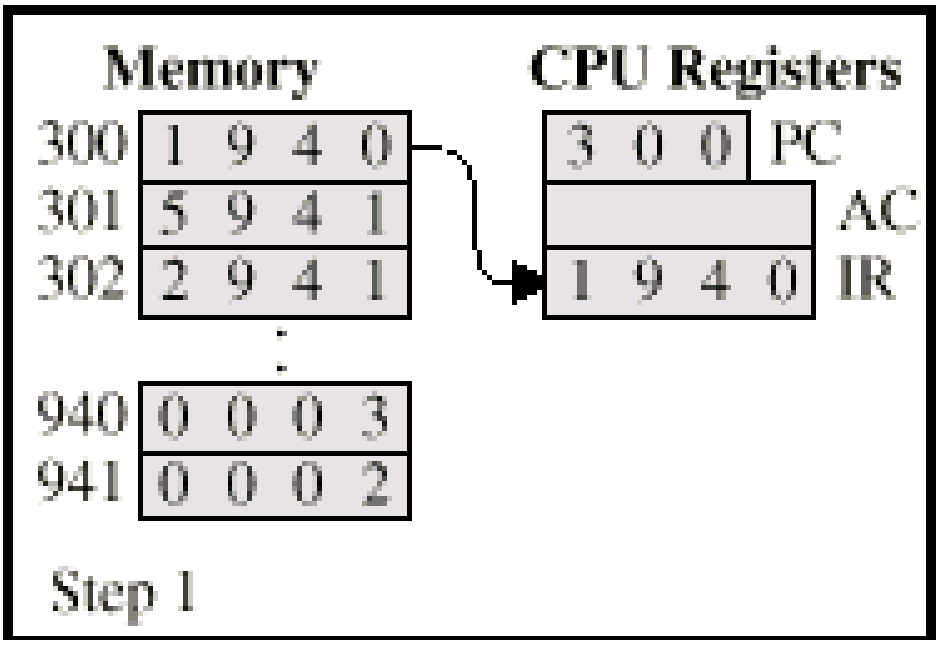
Assembly level organization= Von Neumann

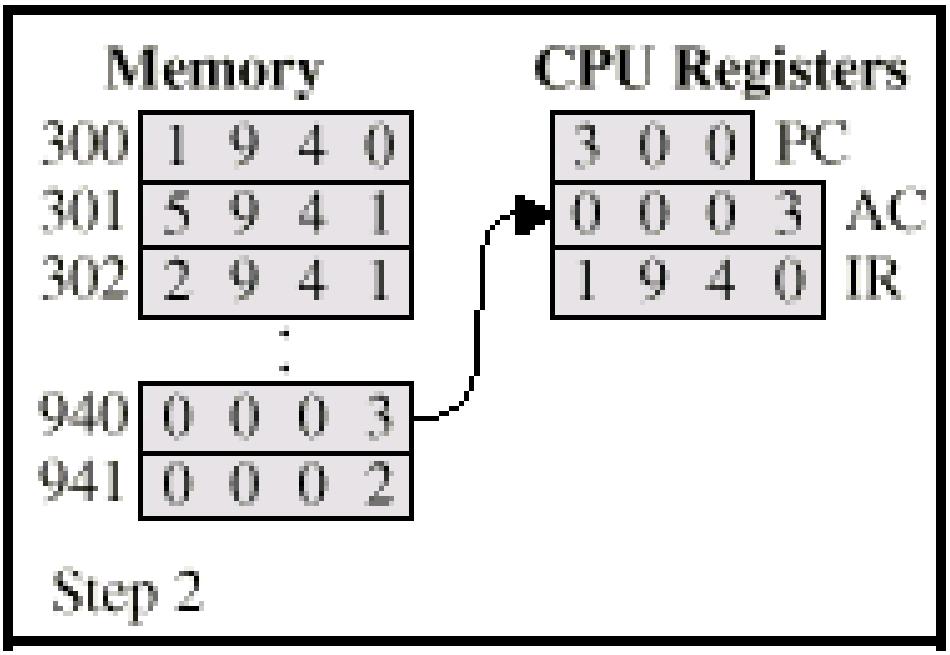
Q6 Trace the Program Execution (**Op-Codes**: 1=load, 2=store, 5=add, …)

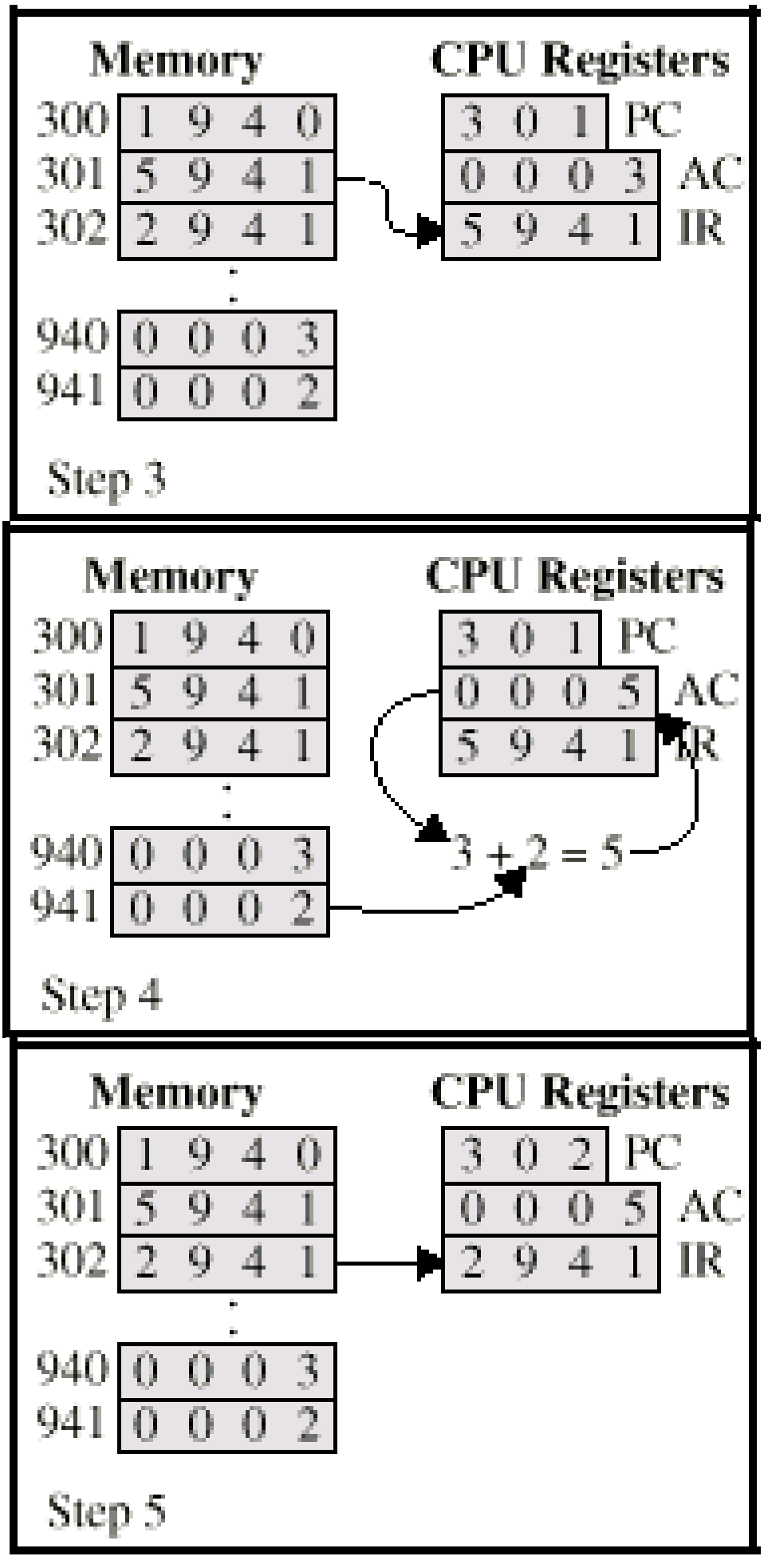
	Memory
300	1 940
301	5 941
302	2 941
	:
940	0003
941	0002
942	0001

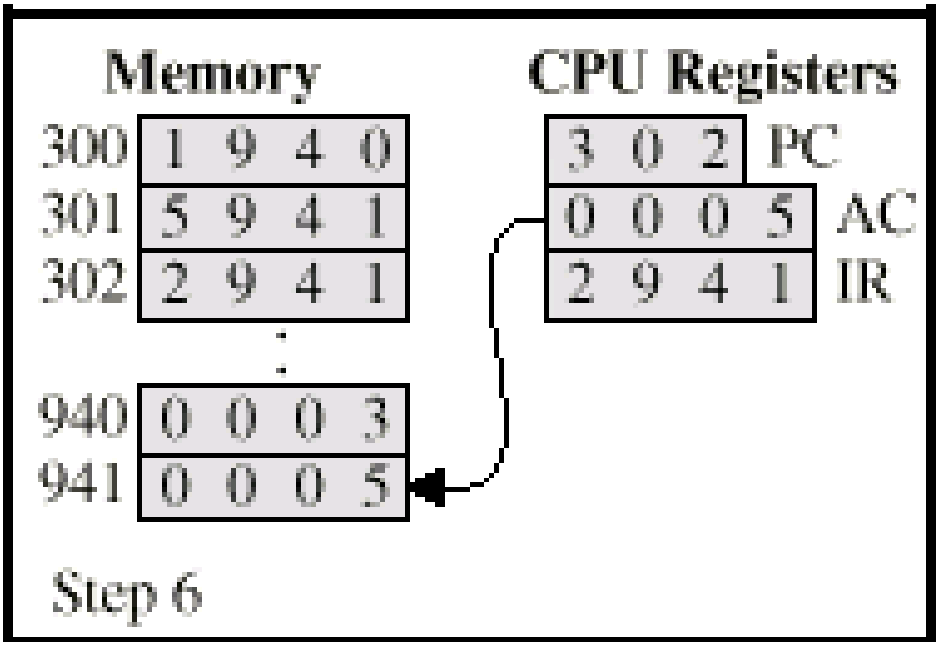
CPU Registers	
	PC
	IR
	AC
	MAR
	MBR

Answer:

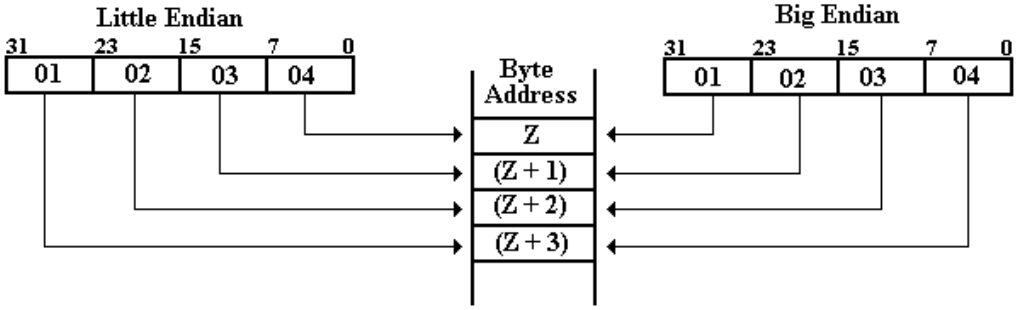





Memory
CPU Registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 2
3 0 1 PC
0 0 0 3 AC
5 9 4 1 IR
Step 3
Memory
CPU Registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 2
3 0 1 PC
0 0 0 5 AC
5 9 4 1 IR
3 + 2 = 5
Step 4
Memory
CPU Registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 2
3 0 2 PC
0 0 0 5 AC
2 9 4 1 IR
Step 5



Q7: Big-Endian and Little-Endian



There seems to be no advantage of one system over the other. Big–endian seems more natural to most people. Big-endian computers include the IBM 360 series, Motorola 68xxx, and SPARC by Sun.

Little-endian computers include the Intel Pentium and related computers.

The big-endian vs. little-endian debate shows in file structures when computer data are "serialized" –

Little-endian Windows BMP, MS Paintbrush, MS RTF, GIF
Big-endian Adobe Photoshop, JPEG, MacPaint

Lecture 8:Report on modern computing

(٠) Just Add Memory
(١) memcomputing
(٢) the central processing unit
(٣) hard drive
(٤) المقاومة الذاكرية هي مقاومة تعتمد قيمتها على مقدار التيار الذي مرّ عبرها سابقا.
(٥) المكثفة الذاكرية هي مكثفة تعتمد سعتها الحالية على قيمة الشحنة التي خزنت فيها سابقا.
(٦) الملف الذاكري هو ملف يعتمد تحريضه الحالي على قيمة التيار الذي مرّ عبره سابقا.
(٧) memcomputer

ويحصل هذا النقل ذو الخطوتين بالاتجاهين لأن ذاكرة الحاسوب لا تستطيع حاليا معالجة البيانات، ولا تستطيع وحدات المعالجة حفظ البيانات. وهذا توزيع شائع للعمل، وهو يحصل في أفضل الحواسيب التي تقوم بأسرع أنواع

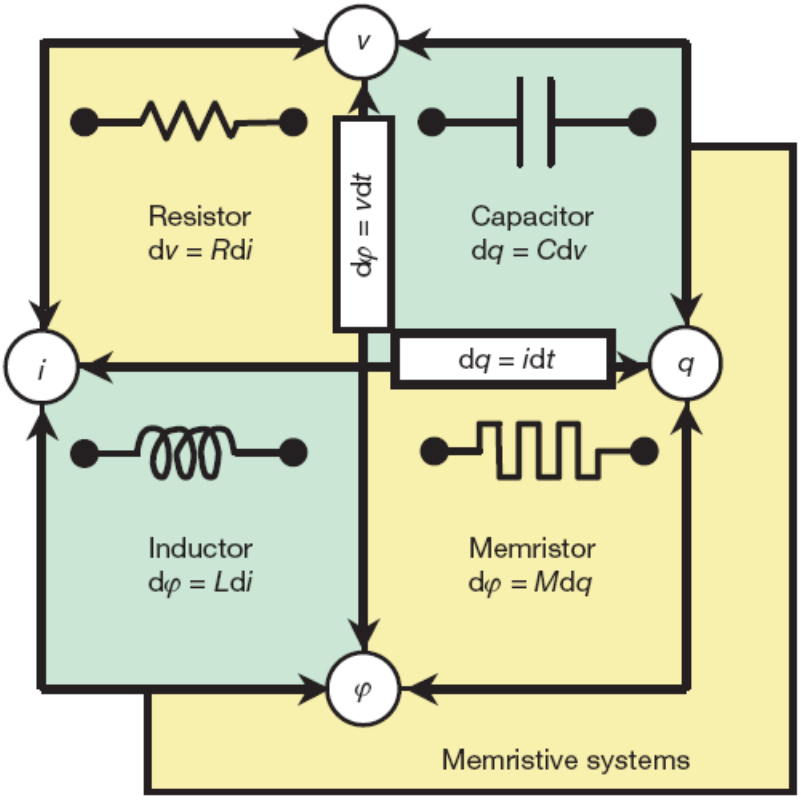
باختصار

في جميع الحواسيب الحديثة تستعمل وحدة للعمليات الحسابية، ووحدة ذاكرة منفصلة توضع فيها البرامج والبيانات. وتستهلك حركة البيانات الكوكبة ذهابا وإيابا بينهما كثيرا من الوقت والطاقة.

لكن تصميما جديدا، يسمى حوسبة ذاكرية[1]، يعمل بطريقة تشبه طريقة عمل العصبونات في الدماغ البشري، حيث توجد كلتا وحدتي الحساب والخزن ضمن الوحدة الفيزيائية نفسها.

ويمكن لهذا أن يعني قفزة كبرى في سرعة الحاسوب وكفاءته، إضافة إلى بنيان حوسبة جديد. لذا يحاول العلماء معرفة أنضل الطرق لاستعمال مكونات الحوسبة الذاكرية المختلفة.

Q8: what are the **four** fundamental two-terminal circuit elements?
Resistor, capacitor, inductor and memristor.



Q9: quantum computing

A Bit encodes a 0 or a 1. A register of *n* bits can store **ANY** *n*-bit number.

A *qubit* (quantum *bit*) encodes 1 and 0 (*superposition*).
A quantum register of *n* qubits stores **ALL** *n*-bit numbers, i.e. 2^n values.

Quantum State $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad a|0\rangle + b|1\rangle = \begin{pmatrix} a \\ b \end{pmatrix}$$

For two qubits,

$$\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}$$

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad |01\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

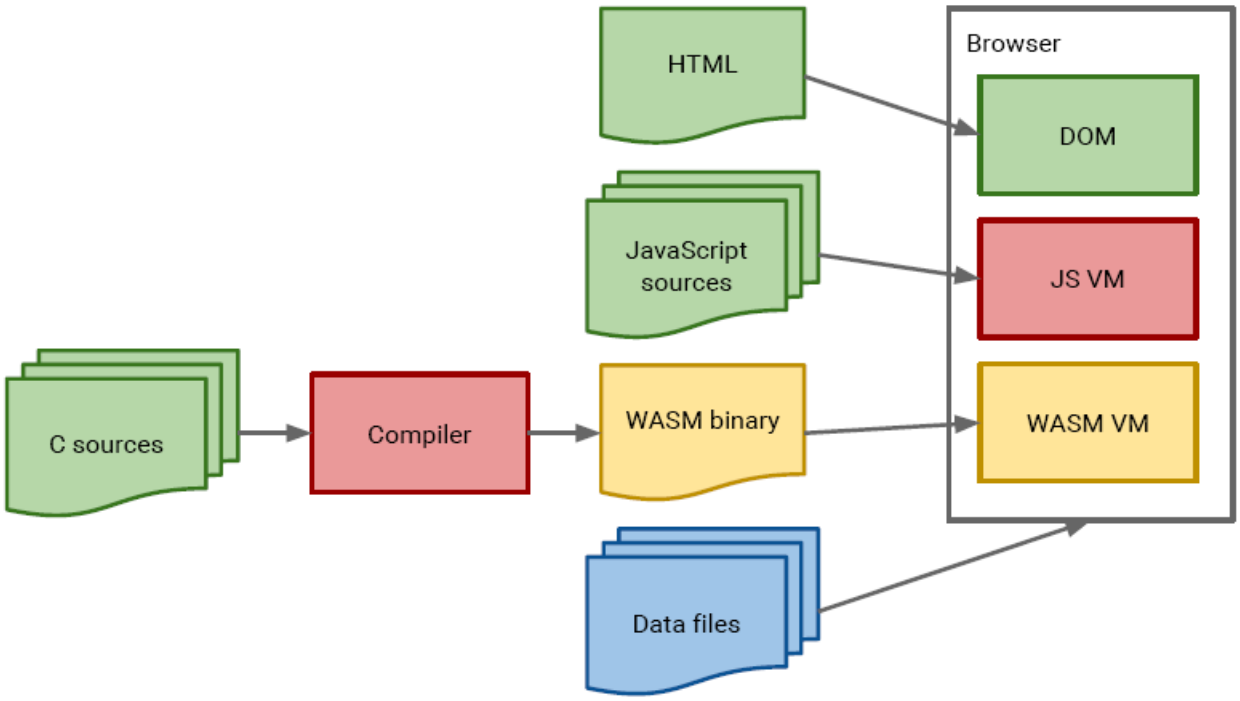
$$|10\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \qquad |11\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

The common 1D quantum gates are:

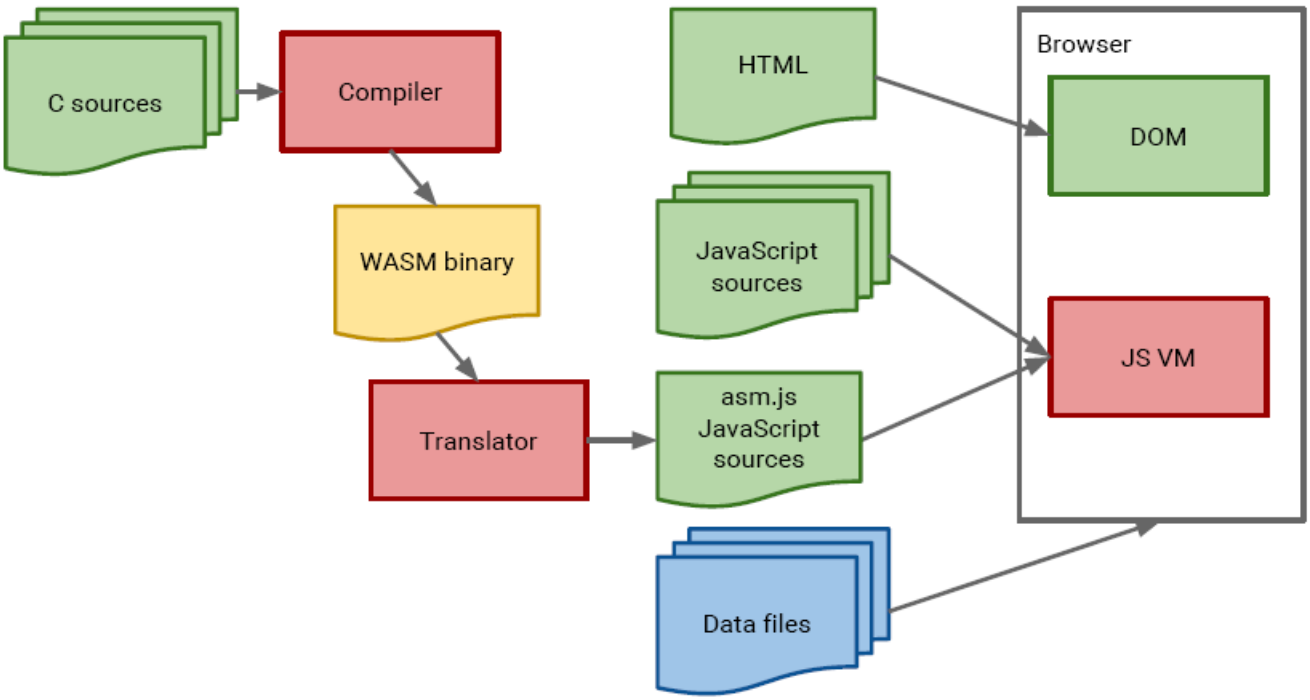
$$Identity = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad NOT = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Phase\_Flip = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \qquad Hadamard = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Q10: draw the architecture of WASM(web assembly) and WASM Polyfill.

WASM



WASM Polyfill



Lecture 9:Macro and Assemblers

Q11 Convert the following C++ code segments into equivalent URM code

a) for(r = 0; r < n; r++) { for_body }

b) if(x==y){if_body} else {else_body}

c) int i ; do{ do_body } while(i != 5);

a)	b)	c)

Z(m+1)
T(1, m+2) // input n
 	
I_l: J(m+1, m+2, I_s)
for_body
S(m+1)
J(1,1, I_l)
I_s:
	T(1, m+1) // input x
	T(2, m+2) // input y
	
	J(m+1, m+2, I_t)
	else_body
	J(1,1, I_s)
	I_t: **if_body**
	I_s:
		Z(m+1) // initially 0
		T(1, m+2) // store 5
		
		I_t: **do_body**
		J(m+1, m+2, I_s)
		J(1,1, I_t)
		I_s:

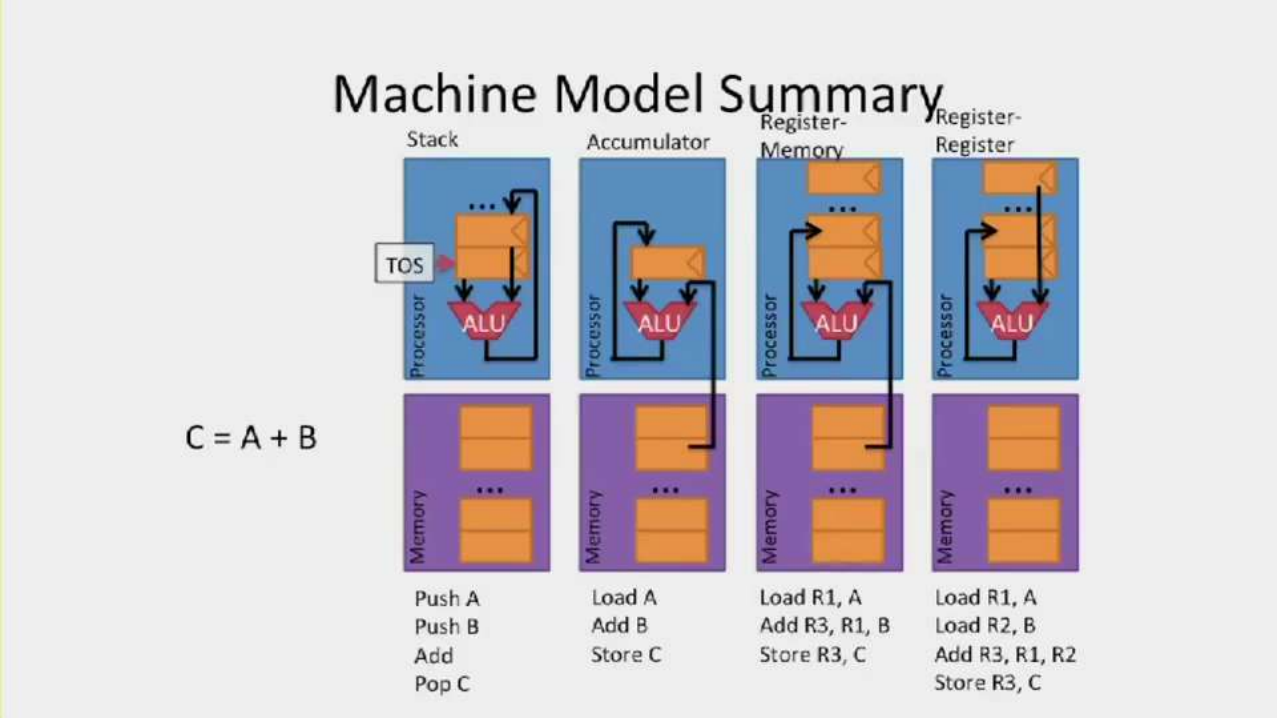
Lecture out: competitive programing / Performance enhancements

Q12 which is faster (x++) or (x+=1) or (x=x+1)?

x++ (inc) faster than x+=1 (add 1) faster than x=x+1 (mov, add, store)

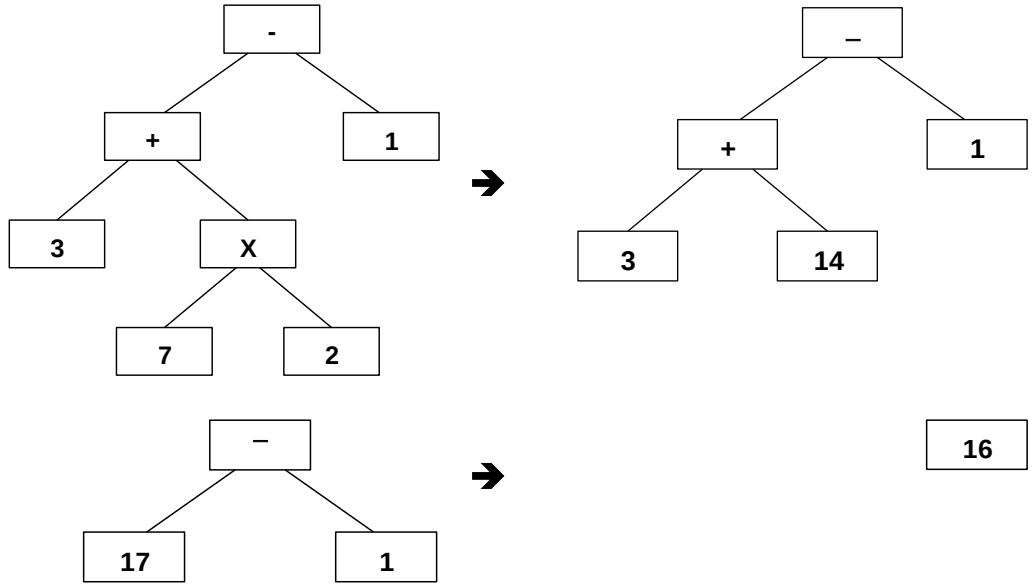
Q13: compare the memory models in calculating C=A+B

Answer



Q14: Convert the expression 3 + 7 × 2 – 1 into Prefix and evaluate using Tree

Expression tree for 3 + 7 × 2 – 1 = - + 3 × 7 2 1



Q15: Convert the expression (A+B)*C into Postfix and evaluate using stack

(A+B)*C AB+C*
infix postfix

Enter values for three variables A=1, B=2 and C=3.

AB+C*	AB+C*	AB+C*	AB+C*	AB+C*
1	2 1	3	3 3	9
push A	push B	pop B pop A push A+B	push C	pop C pop A+B push (A+B)*C