EO107
計算機概論
Introduction To Computer Science
Semester 102-1

白小明
Jonathon David White
元智大学光电系
R70740, R70723
WhiteJD@XiaoTu.com

修改:中華民國 102 年 11 月 5 日 14 時 14 分 39 秒
Modified AD13 年 11 月 5 日 (Rev 8686)

"To Him Who Is Above And Beyond All"

Table of Contents

Introduction To Computer Science

J D White,

1. Introduction

1.1 Facilitators

a. Lecturer: 白小明 小明白

1. Background: http://www.xiaotu.com/whitejd/per/index.htm

Jonathon David White was born in Oakville, Canada but has since lived in many other countries. Even during his undergraduate days at McMaster University, he already had a cosmopolitan outlook on life, being active in the Chinese Christian Fellowship. After obtaining his Ph.D., also from McMaster University, he worked and taught in China, Japan, and Taiwan – where he met and married Wu Xiuman – and then Malaysia at Multimedia University. After 4 years (1999-2003) in the Faculty of Engineering and Technology at the Melaka campus of Multimedia University, he moved with his family to Taiwan. He is now Associate Professor at Yuan Ze University. He and his wife have two daughters, Ai-en (Charity Grace) and Liang-En (Ruth Ann) as well as two sons, You-en (Johann Donald) and Li-En (Leon Joshua).

In contrast to Dr. Chai's formal education in Computer Science, Dr. White's experience in programming has largely been self-taught on a "need-to-know" basis. His introduction to ANSI-C came in 1994, when he took a position in the Ocean Remote Sensing Institute in Qingdao, China. Upon arrival, he was given a book introducing ANSI-C (in Chinese) and told to interface a computer, laser and detector – allowing him to simultaneously learn ANSI-C and Chinese! This "need-to-know" has resulted in the the method of teaching of this course.

Reflecting back, probably the most important decision made by Jonathon was that to follow Jesus Christ. It is only in the light of this decision and the guidance received from God that one can understand this life's trajectory.

2. Family: 爱有力量 https://www.youtube.com/watch?v=G1h9AhUh7o8
3. Research: http://www.xiaotu.com
4. Email: whitejd@xiaotu.com
5. Calendar: http://www.xiaotu.com
6. Office: R70740, R70723 & Lab
7. Office Hours: Tuesdays and Thursdays, 11AM to 12 noon

b. Teaching Assistants

1. Kevin: Vietnamese Ph.D. student R70740
2. Aray: Taiwanese Ph.D. student R70740

1.2 Respect

a. Classroom Expectations

1. Arrive on Time (after attendance deemed absent)
2. Listen to Lectures
3. Ask Questions (bonus marks)
4. Listen to fellow students
5. Food and Drinks are OK in the classroom
6. Do not leave garbage in classroom
7. During class: (as this distracts other students)
   - i. No FACEBOOK,
   - ii. No computer games
   - iii. No checking email
   - iv. No videos

Students disobeying rules will be asked to leave the classroom. If cited more than three (3) times, student will be asked to drop the course.

b. Rules for the Computer Room 電腦教室使用規定

1. 上課注意事項：
   i. 準時到教室，遲到禁止進入教室。
   ii. 在教室裡請勿飲食，食物和飲料禁止帶進室內。
   iii. 每位同學上課都有固定位置，點名前請勿隨意更換位子。
   iv. 請勿隨意更動教室內電腦設定。
2. 下課注意要點：
   i. 請將垃圾帶走，丟在安全門外的垃圾桶。
   ii. 請將座椅歸回原本的位置。
   iii. 每個禮拜會安排值日生在課後檢查教室，請務必配合。
   iv. 有違反規定的將登記扣分
   v. 以上如有不清楚的部份，請找老師或助教協助

1.3 Course Overview

This course is the first in a series of three courses for Optics students dealing with computer programming. The goals for this first course are twofold. First, for this first course is to have students understand the fundamental knowledge of computer science including the history of computers, representation of information, hardware components, programming concepts, role of operating systems, and status of networking communication. Second, students will learn how to write, edit and execute a Simple ANSI-C program. (本課程目標在於使學生具備計算機相關基本知識, 包括電腦發展史, 資訊表達法, 硬體組成, 程式設計概念,作業系統角色, 與網路通訊發展概況等, 以橋接通訊領域相關之進階課程.在程式設計概念部分將教授 C 與語言之基本知識，以利學生在修習高階課程時具有基本程式設計之能力。)

The teaching format is lectures followed by small groups(3-4 students) completing a worksheet with help from the teacher and Tas. At certain points in the course we will make use of the computer room to run code.

Table: Key Topics in this course

Topic	題目
(Theory of Computing): Finite State Automata and Turing Machines	電腦理論
Algorithms and UML (Universal Modeling Language)	演算法和 UML
Number Systems and Data Storage	數值系統和資料儲存
Networks and Operating Systems	網路與作業系統
Computer Organization -- Von-Neumann Architecture	計算機組織
Coding: Machine --> Assembly --> ANSI-C	簡易程式設計

1.4 Textbooks/References

We will be using selected chapters from the following two textbooks in this course.

1. Ian Chai and Jonathon White, *Structuring Data and Building Algorithms: An ANSI-C Based Approach*, McGraw-Hill (@ Caves, Contact: Tel : 02-23113000#212 / Fax : 02-2388-8822 at McGraw-Hill) ISBN: 978-0071271882 Chapters 1, 7, 11, 12 (in eo109: 1, 2, 7)
2. Behrouz A. Forouzan, *Foundations of Computer Science*, Cengage Learning EMEA; 2 edition (December 5, 2007) ISBN 978-1844807000, Chapters 2, 3, 4, 5, 6, 7, 8. Ref: Appendix B

1.5 Key Websites

1. http://www.xiaotu.com/tea/yzueo109.htm (Animations for this class)
2. http://www.sdba.info (Textbook Animations for FSA and Turing Machines)

The first textbook will be used for all three courses in computer science offered by our department. Key concepts are covered in the animations and view graphs

1.6 Course Delivery and Milestones

For this course the progress of students is monitored through a series of milestones. Figure 1 shows the topics to be studied in this course and their relationship, along with the key milestones in terms of a modified UML diagram. In this diagram milestones are marked by diamonds.

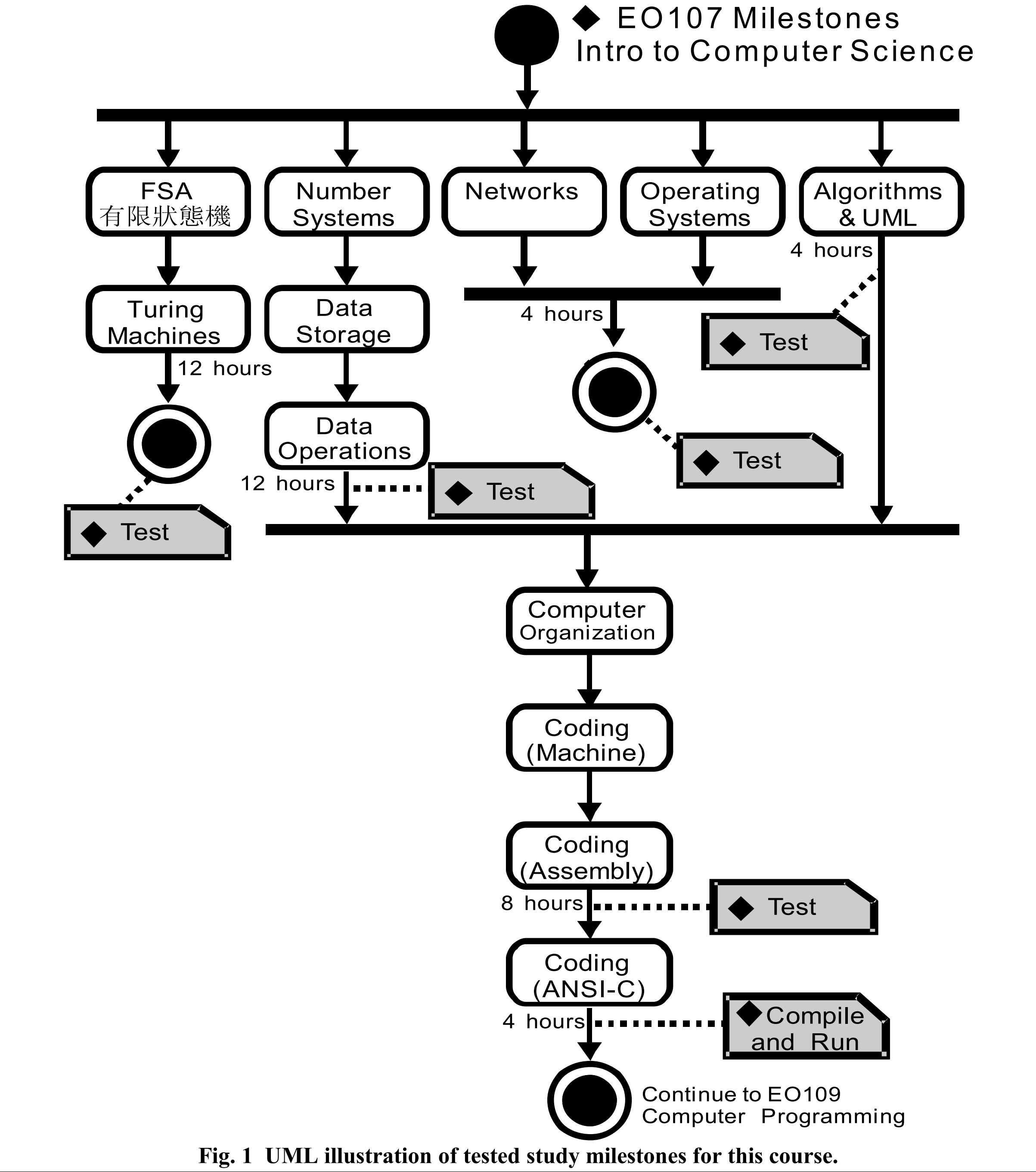


Fig. 1 UML illustration of tested study milestones for this course.

In the above diagram the relationships between topics can also be seen. For example, in other to understand Turing Machines, one must first understand FSA's. Topics such as Operating Systems and Networks are independent – one does not need any previous knowledge to understand these topics and failure to understand these topics will not hinder one's further understanding in the course.

Fig. 1 also contains information about the number of lecture/contact hours will be spent in studying each of these topics. For example, 12 hours will be spent studying FSA's and Turing Machines, after which students progress will be evaluated by means of a short test.

1.7 Grading

As can be seen in Fig. 1, there are a total of six key milestones at which on is evaluated. The percentage each milestone contributes to the final mark is directly proportional to the number of hours assigned this topic with each hour of study being awarded 2 points in the final evaluation. For example, since twelve (12) hours are spent studying FSA's and Turing Machines, this milestone is worth twenty-four (24) points in the final evaluation.

Table 1: Milestones and Their Weight for Midterm and Final Assessments

項目編號	項目名稱 Milestone	期中評量權重 Midterm	學期總成績權重 Final
1	Milestone: Computer Theory	48%	24%
2	Milestone: Algorithms & UML	16%	8%
3	Milestone: Networks and OS	16%	8%
4	Milestone: Number Systems and Data Storage	0%	24%
5	Milestone: Computer Architecture and Coding	0%	16%
6	Milestone: First Program in ANSI-C	0%	8%
0	Attendance, Tutorials and Small Group work	20%	12%
0	BONUS: Successful Group Leaders, Pointing out errors	max 5%	max 5%

Unlike other courses, each milestone is evaluated no a Pass/Fail basis and each student can try the test as many times as are required to pass the milestone. If one passes the milestone at one's first attempt, one receives the full point score for the milestone. If, however; requires a second attempt to pass the milestone, then only 85% of the marks assigned that milestone will be awarded. The calculation of marks for this course is summarized in Fig. 2.

Bonus marks are available for pointing out errors and mistakes in the teacher's lecture materials. Each mistake will give the first student who points it out an additional 1 point. Each student can earn a maximum of 5 points for finding errors in the teacher's lectures.

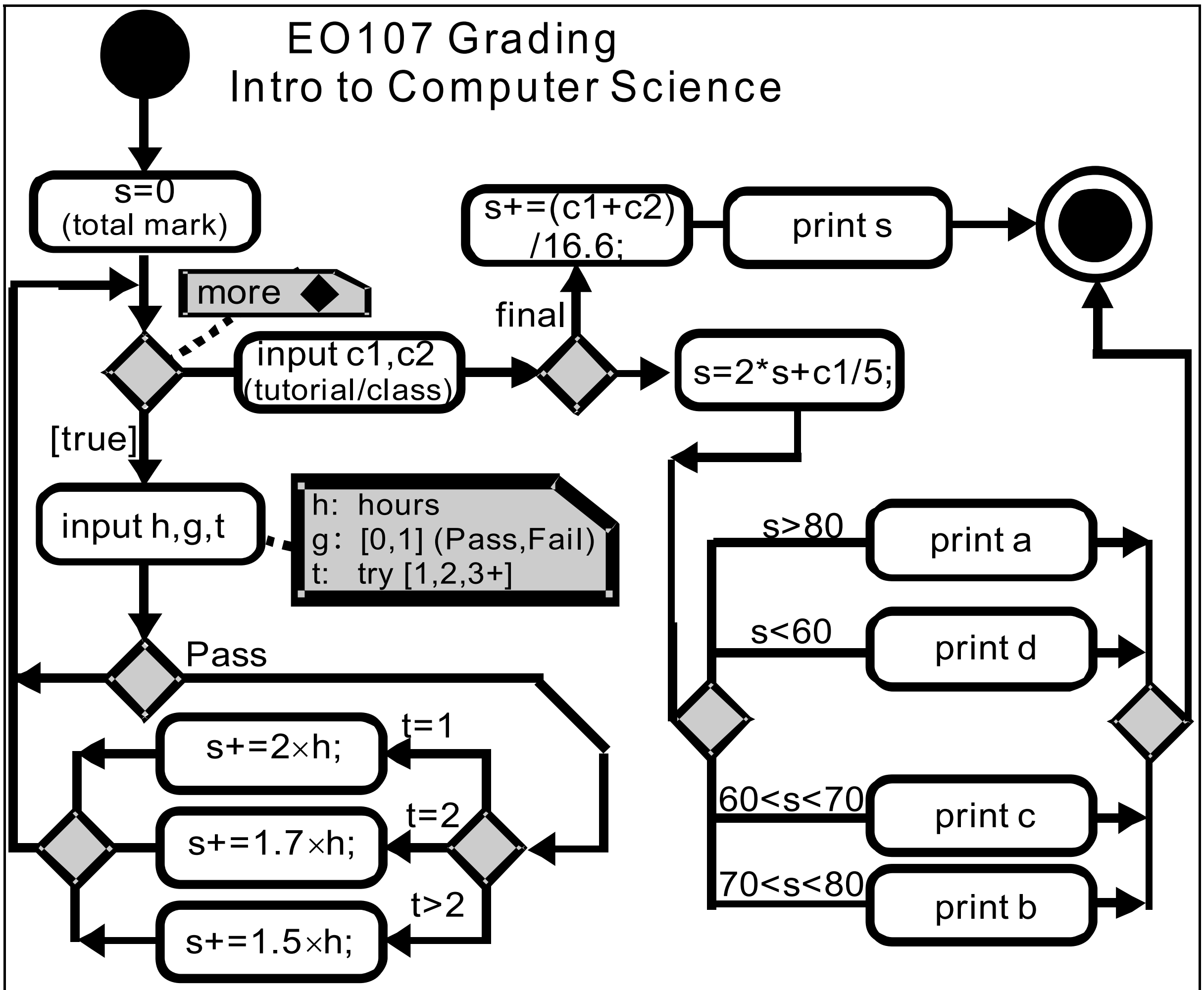


Fig. 2 UML representing Grading Procedure

For example, if a student named 小明, passes milestones 1 and 2 on his 1st try, fails to pass milestone 3, passes milestone 4 on his 2nd try, and fails to pass milestone 5 and has excellent class performance, his final grade would be : s = (2*12 + 2*4 + 0 + 1.5*12 + 0 + 0) + 12 = 62%. Since milestone 6 is dependent on milestone 5 (see Fig. 1), failure to pass milestone 5 means that 小明 is not eligible to take the test for milestone 6. Since their are no other milestone 4,5,6 are not dependent on milestone 3, failure to pass this milestone does not affect 小明 eligibility to pass the future milestones.

1.8 Calendar

Class	Topic	Wk	Date
1	Welcome	1	09.17 @ 18:30
		1	09.20 @ 13:00
		2	09.24 @ 09:00
2	Theory of Computing	2	09.27 @ 13:00
3	Theory of Computing	3	10.01 @ 09:00
4	Theory of Computing	3	10.04 @ 13:00
5	Theory of Computing	4	10.08 @ 09:00
6	Theory of Computing	4	10.11 @ 13:00

Class	Topic	Wk	Date
7♦	Theory of Computing	5	10.15 @ 09:00
8	Algorithms & UML	5	10.18 @ 13:00
		6	10.22 @ 09:00
9♦	Algorithms & UML	6	10.25 @ 13:00
10	Networks and OS	7	10.29 @ 09:00
11♦	Networks and OS	7	11.01 @ 13:00
12	Number Systems and Data Handling	8	11.05 @ 09:00
13	Number Systems and Data Handling	8	11.08 @ 13:00
14♦	Missed Milestones Retest (2nd attempt)	9	11.12 @ 09:00
♦	Missed Milestones Retest (3rd attempt)	9	11.15 @ 13:00
		A	11.19 @ 09:00
15	Number Systems and Data Handling	A	11.22 @ 13:00
16	Number Systems and Data Handling	B	11.26 @ 09:00
17	Number Systems and Data Handling	B	11.29 @ 13:00
18♦	Number Systems and Data Handling	C	12.03 @ 09:00
19	Computer Organization and Machine Language	C	12.06 @ 13:00
20	Computer Organization and Machine Language	D	12.10 @ 09:00
21	Computer Organization and Machine Language	D	12.13 @ 13:00
22♦	Computer Organization and Machine Language	E	12.17 @ 09:00
23	JEdit & My 1st C-program	E	12.20 @ 13:00
24♦	JEdit & My 1st C-program	F	12.24 @ 09:00
25♦	Retry Milestones	F	12.27 @ 13:00
		G	12.31 @ 09:00
		G	01.03 @ 13:00
		H	01.07 @ 09:00
		H	01.10 @ 13:00
		I	01.14 @ 09:00
		I	01.17 @ 13:00

1.9 Detailed Lecture Plan / Teaching Schedule with References

a. Welcome (*2hrs)*

1. Teacher Introduction **(eo107-1.ppt)** *(101-1wk1)*
2. Textbooks & Notes
3. Evaluation
4. Division into small teams (3 to 4 students/team, choose group leader)

5. Notation
   i. *Underlined Blue Italic Arial is the expected time for an item*
   ii. **Bold green underlined References: textbook, view graphs or animation**
   iii. *dashed underlined pink indicates predicted break points in the lecture series*
   iv. double underlined black is tutorial

b. Theory of Computing *(12 hrs + 1 hr milestone)*
1. Finite State Automata **(Chai, pgs. 369-384)**
   i. Presenting the crossing river problem
   ii. Theory of FSA **(Chai pgs. 369-374, b0fsaint.pps)**
   iii. Light Bulb Example
   iv. Example: Even Number of 1s checker
   v. Group work: Even number of 1s and Even number of 0s checker *(101-1wk1)*
   vi. Solving the River Crossing Problem **(Chai pgs. 374-378, b3mantig.pps)** *(101-1wk2)*
   vii. Group work: Model an ATM (Automatic Banking Machine)
   viii. Online Paper: http://www.enel.ucalgary.ca/People/wangyx/Courses/SENG523/Tutorials/ATM%20Architecture.pdf
   ix. Build FSA Online (DEMO)
   x. RE and Non-Deterministic FSAs **(Chai pgs. 379-384, b4renfsa.pps)**
   xi. Small Group Tutorial (50 minutes): (FSA#1) *(101-1wk2)*
   xii. Run selected FSAs Online *(101-1wk3)*
   xiii. Take up Tutorial (FSA#1)
   xiv. Homework (FSA#2) **(Chai pgs. 384-85, all questions)**
2. Turing Machines **(Chai pgs. 385-397, c-turing.pps + Chai links)**
   i. Limitations of FSAs
   ii. Details of a TM
   iii. Sample TMs
      a. Duplicator
      b. Add 1
      c. Incrementer
      d. Number of 'a' = number of 'b' checker
   iv. Real Turing Machine: http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/032610-diy-turing-machine
   v. Introduction of Computer Room
   vi. Review of Lab Rules **(eo107-1.ppt)** *(101-1wk3-end)*
3. General Purpose Turing Machine *(101-1wk5-start)*
      a. Take up Homework FSA#2 + demo with Online-FSA **(Chai pgs 384-85, all questions)**
      b. Review Turing Machine Slides
      c. Demonstration of our Turing Machine Simulator with various TM and tapes
      d. Small Groups, Tutorial (90 minutes) Turing Machine: Subtract 1 & Turing Machine Add 2 *(101-1wk5-end)*
      e. Group Demonstration of their controller and tape (40') *(101-1wk6-start)*
4. A Practical Computer:
   i. The Von Neumann Model
      a. Introduction **(Forouzan, ch 1.2)**
      b. Computational Philosophy
         -Change A to B http://www.sdba.info/theory/c3btoa.swf
         -Add 1 http://www.sdba.info/theory/add1.swf
   ii. Computer Components **(Forouzan, ch 1.3)**

iii. History **(Forouzan, ch 1.4)**

5. Other **(Forouzan, ch 1.5)**
   i. Social and Ethical Issues
   ii. Computer Science as a Discipline *(101-1wk6-end)*
6. **MILESTONE #1 THEORY OF COMPUTING**

c. Algorithms & UML *(4 hrs)*

1. Sample Algorithms:
   i. Baking Muffins
   ii. Dictionary Search for a word (linear-sequential vs binary)
2. Algorithms (**Chai Ch 7**) *101-1wk15start*
   i. Language Independence: **Chai-71langua.pps** *T=1 hr*
   ii. Quality Concerns **Chai-72qualiti.pps**
   iii. Time Complexity **Chai-73timeco.pps (Part 1)**
   iv. Error Propagation **Chai-75errorp.pps**
3. Expressing Algorithms in UML activity diagrams (already seen state diagrams)
   i. Symbols in UML activity diagrams *(inserted in 102-1 year)* **eo107...algo.swf**
   ii. Symbolizing types of flow in UML **eo109...6flow.swf**
      a. linear flow
      b. repetition
      c. conditions
4. Expressing an algorithm in UML (Baking Muffins)
5. Tutorial: Algorithms and UML
   i. Time to do as groups
   ii. Take up together
6. **MILESTONE #2 ALGORITHMS AND UML**

d. Networks and Operating Systems *(4 hrs)*

1. Computer Networks **Forouzan Ch 6**
   i. Overview
      a. Criteria
      b. Structures
      c. Categories
      d. an internet
   ii. The Internet -- A Layered Structure http://www.xiaotu.com/tea/yzueo107/anetwork.swf
      a. Application Layer (eg. www, telnet, telephony)
      b. Transport Layer
         -UDP vs TCP : http://www.skullbox.net/tcpudp.php
         -UDP vs TCP: http://www.youtube.com/watch?v=KSJu5FqwEMM
         -SCTP
      c. Network Layer
      d. Data Layer
      e. Physical Layer *101-1wk13end*
   iii. Review http://www.xiaotu.com/tea/yzueo107/anetwork.swf *101-1wk14start*
2. Operating Systems *101-1wk14start*
   i. Overview including diners problem http://www.xiaotu.com/tea/yzueo107/aos.swf
   ii. In detail **Forouzan Ch 7** *101-1wk14end*
3. **MILESTONE #3 NETWORKING AND OPERATING SYSTEMS**

e. Number Systems and Data Handling *(12 hrs)*

1. Number Systems
   i. Why need to discuss? Data Storage is not base 10 it is base 2

ii. Number systems in general – see workbook

iii. Non-Positional Number Systems http://www.xiaotu.com/tea/yzueo107/num_zh.swf

a. Roman
b. Chinese

iv. Positional Number Systems: Representation :

a. Introduction: Course Notes
b. Examples **Forouzan ch 2.2**

v. Conversion between bases

a. Course Notes for examples and UML
b. Hand calculation for Base 3 to Base 10 and back.
c. Numberous Examples **Forouzan ch 2.2**
d. http://www.xiaotu.com/tea/yzueo107/numsys.swf
e. http://www.xiaotu.com/tea/yzueo107/convert.exe
f. Tutorial: Number System Conversion (60 minutes) *(101-1wk6-end)*

vi. Review and Lessons Learned *(101-1wk7-start)*

a. Review:
-http://www.xiaotu.com/tea/yzueo107/num_zh.swf
-http://www.xiaotu.com/tea/yzueo107/numsys.swf
-http://www.xiaotu.com/tea/yzueo107/convert.exe
b. Take up Tutorial Number System Conversion
c. Key Point: Loss of Precision in conversion for Real Numbers (Decimal Points) http://www.xiaotu.com/tea/yzueo107/num_err.swf
d. Impacts on how we store our data.

2. Data Storage **(Chai Ch 1.1, Forouzan Ch. 3)**

i. Data Types (storing different types) **Forouzan Ch. 3.1**

ii. Variables (Chai) http://www.xiaotu.com/tea/yzueo107/datastr.swf *(`01-1wk8-start)*

iii. Storing Numbers-Intro **Chai-12 store3.pps (consider eo107 notes)** *(101-1wk7-end)*

iv. Storing Numbers-Fixed vs Floating-Point, 2s comp and IEEE

a. Key Ideas http://www.xiaotu.com/tea/yzueo107/datastr.swf
b. Examples - **Forouzan Ch. 3.2,**
c. Tutorial: Storing Data: Fixed and Floating-Point (60 minutes)

v. Other Types of Data **Forouzan Ch. 3.3,**

a. Storing Text
b. Storing Audio
c. Storing Images
d. Storing Video

vi. Tutorial: Storing Data: Text (See Appendix B: ASCII)

vii. Review http://www.xiaotu.com/tea/yzueo107/2review.swf *(101-1wk8-end)*

3. Operations on Data

i. Logic: How to then application http://www.xiaotu.com/tea/yzueo107/abitoper.swf , **Forouzan Ch 4.1**

a. NOT : (Unitary) Complementing a Bit Pattern (A to $\overline{A}$)
b. AND : Unset Bits (Force Bit to Zero)
c. OR : Set Bits (Force Bits to One)
d. XOR : Flip Specific Bits
e. Tutorial Bit Operations: Logic & Shift Demo question (a)

ii. Shift http://www.xiaotu.com/tea/yzueo107/abitoper.swf **Forouzan Ch 4.2**

a. Logical
b. Arithmetic
c. Tutorial Bit Operations: Logic & Shift Demo question (b) as example

iii. Complete Tutorial Bit Operations: Logic & Shift (30 min)
iv. Arithmetic (Addition/Subtraction)
a. Introduction to calculations (**Chai**) **Chai-12calcu4.pps**
b. Choosing the Containor (**Chai**) **Chai-12exact5.pps**
c. By hand (Subtraction is adding negative number!)... **Chai**
-base-10
-base-2 (http://www.xiaotu.com/tea/yzueo107/addfix.swf)
-Link to Animation http://www.is.wayne.edu/drbowen/casw01/AnimAdd.htm
d. S & M notation
-Explanation http://www.xiaotu.com/tea/yzueo107/addfix.swf
-Examples **Forouzan Ch 4.3**
-Tutorial Binary Arithmetic: Fixed-Point
http://www.xiaotu.com/tea/yzueo107/addfix.swf Demo Question (a)
e. Two complement notation
-Explanation http://www.xiaotu.com/tea/yzueo107/addfix.swf
-Examples **Forouzan Ch 4.3**
-Tutorial Binary Arithmetic: Fixed-Point
http://www.xiaotu.com/tea/yzueo107/addfix.swf Demo Question (b)
v. Tutorial Binary Arithmetic: Fixed-Point *(101-1wk10-end)*
vi. Review fixed-point http://www.xiaotu.com/tea/yzueo107/addfix.swf
vii. Take-up Tutorial Binary Arithmetic: Fixed-Point*(101-1wk11-start)*
viii. Real Numbers
a. Explanation http://www.xiaotu.com/tea/yzueo107/addfloat.swf (Part 1)
b. UML **Forouzan Ch 4.3**
c. Examples **Forouzan Ch 4.3**
d. Tutorial Binary Arithmetic: Floating-Point
http://www.xiaotu.com/tea/yzueo107/addfloat.swf (Part 1 Example)
ix. Tutorial Binary Arithmetic: Floating-Point
4. **MILESTONE #4 DATA OPERATIONS**

f. Computer Organization *(8 hrs)*
1. Overview of Computer http://www.xiaotu.com/tea/yzueo107/comporg.swf (Scene: Visual Summary)
i. CPU **Forouzan Ch 5.1**
ii. Main Memory **Forouzan Ch 5.2** eg. Maximum memory for 32-bit addressing with 32-bit word is 16GB
iii. I/O System **Forouzan Ch 5.3**
iv. System Interconnect
a. http://www.xiaotu.com/tea/yzueo107/comporg.swf (Scene: Parallel/Serial)
b. **Forouzan Ch 5.4,**
2. Review http://www.xiaotu.com/tea/yzueo107/comporg.swf (Scene: Visual Summary)
3. Program Execution: Machine Cycles **Forouzan Ch 5.5**
4. Architecture: CISC/RISC + Parallel Processing, Pipe-lining **Forouzan Ch 5.6** *101-1wk11-end*
5. Example of Simple Computer...**Forouzan Ch 5.7** *101-1wk12-start*
6. Tutorial: Assembly Language Programming (Appendix A: ASS Assembly Language) program (a and b).
7. Take up Tutorial Assembly Language Programming *101-1wk13start*
i. http://www.xiaotu.com/tea/yzueo107/comporg.swf Scene: ProgramRunning(2s Comp)
ii. blackboard for Assembly Language Programming question c
iii. Tutorial: Algorithms: From Concept → UML → ASS (Homework)

iv. Take up Tutorial: Algorithms: From Concept → UML → ASS

8. **MILESTONE #5 COMPUTER ORGANIZATION AND ASSEMBLY PROGRAMMING**

g. JEdit & My 1st C-program. *(4 hrs)*

1. Computer Lab Introduction (6. Rules for Computer Lab) *101-1wk15end*
2. programming flow http://www.xiaotu.com/tea/yzueo107/2prog1.swf
   i. editing
   ii. compiling
   iii. linking
   iv. running
   v. debugging *101-1wk16end*
3. Tutorial: Simple ANSI-C Program
4. **MILESTONE #6 COMPUTER PROGRAM RUNNING**
   i. Time in Computer lab to practice (1 hr)
   ii. Examination in groups (2 hrs)
   iii. Simple ANSI-C Program http://www.xiaotu.com/tea/yzueo107/aqfin_lab.swf

2. Theory of computing

2.1 Finite State Automata (FSA)

a. Definition

An FSA is defined by a tuple (Q, Σ, q_o, F, T): – Q : set of states – Σ : set of inputs symbols (alphabet) – q_o : starting state – F : set of final state – T : transition functions An FSA can be represented – graphically by state diagrams – compactly with a transition table or transition functions	一個有線狀態機定義一組變數(Q, Σ, q_o, F, T) – Q : 狀態設定 – Σ : 輸入符號的設定 (符號系統) – q_o :開始狀態 – F : 結束狀態 – T : 轉換函式 一個有線狀態機能被表示成 – 圖形狀態圖 – 簡潔的轉換功能或是轉換表

b. Example: Light Bulb and Switch

A simple example of an FSA is the combination of a light bulb and a switch. For this simple system one can specify:

– Q = {OFF, ON}
– Σ = {P} where P stands for "Push the switch"
– q_o={OFF}
– F ={OFF}

The transition function can be respresented in either of the three forms:

State \ Input	P
ON	OFF
OFF	ON

T(OFF,P) → ON
T(ON,P) → OFF
or

```
[OFF,P, ON]
[ON ,P,OFF]
```

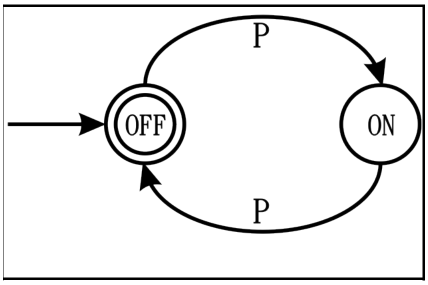


Fig. {*fsa-example*} Equivalent Representations of the Transition Function (a) transition matrix (b) transition function (c) state diagram

Note that the state diagram visually shows the starting and final accept states but for the other representations, these need to be specified independently.

c. The Language of an FSA: Regular Expressions

Regular expressions allow the language that an FSA accepts to expressed compactly. In addition to the alphabet (Σ) used by the FSA, regular expressions (RE) make use of the following symbols:

Symbol	English	Chinese
*	zero or more of the previous group	前面的字串零個以上(包含零個)
|	either previous or next group	包含前面或後面的字串
()	groups elements between the parentheses	在括號內的字串
+	one or more of the previous group	前面的字串一個以上(包含一個)
ε	empty string (NFSAs only)	空白字串

For example, the RE representing the light bulb in the previous example is `(PP)*`.

d. Deterministic and Non-Deterministic FSAs

Finite State Automatas can be divided into two types: deterministic (DFSA) and non-deterministic machines (NFSA). The former is a subset of the latter. The differences are summarized in the following Table {*fsa-nfsa_dfsa*}. Anything that can be expressed using a NFSA can also be expressed using a DFSA.

DFSA	NFSA
Transitions (T) must be specified for all input symbols (Σ) for every state (Q)	Not necessary to specify transitions (T) for all input symbols (Σ) for every state (Q)
Only one transition (T) may exist for each input symbol (Σ).	Multiple transitions (T) may exist for one input symbol (Σ)
Any change in state (Q) requires an input symbol (Σ).	You can change states (Q) without using an input symbol (ε)

Table {*fsa-nfsa_dfsa*} Deterministic and Non-Deterministic Finite State Automata

2.2 Turing Machines

a. Introduction

Unfortunately an FSA is insufficient to model a complete computer. In order to do this we need a Turing Machine. A Turing Machine is comprised of four components:

- Tape (infinitely long, divided into cells with each one holding a symbol from the alphabet or a blank symbol)
- Head (reads or writes information to the tape and can move the tape left or right by one cell)
- Controller (Modified FSA that specifies for a given state and input symbol, a symbol to write on the tape, the direction to move the tape and a new state for the machine.
- Register (stores the current state of the machine)

The controller can be respresented in one of the following three forms:

- State Diagram
- Transition Table
- Set of Quintets (Current State, Input Symbol, Write, Move Tape, New State)

While a table is commonly used for computer programming, the state diagram representation is often used for human usage.

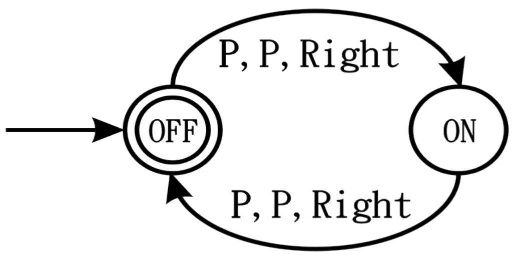
b. Example: Light Bulb and Switch: Automatic Time setting.

If the machine is set to read and write one symbol per hour, we can specify the time of the day to turn a light on off when we leave the house. . For symplicity lets assume that we want our light to go on at 9AM in the morning, off at lunch time (12 noon), on at 1PM and finally off at 5PM in the each day. We could then use the following tape:

									P			P	P					P					

Assuming our head starts at the far right of the tape, and the initial state q_o={OFF} We could then write our controller either as a set of Quintets as where the first column is the current state, the second column is the read symbol, the third column is the symbol to write, the fourth column is the direction to move the tape and the final column is the new state or a state diagram (humans).

```
[OFF, , ,Right,OFF]
[ON , , ,Right,ON ]
[ON ,P,P,Right,OFF]
[OFF,P,P,Right,ON ]
```



3. Algorithms and UML Activity diagrams

3.1 Algorithms

An algorithm describes the steps that one needs to take to take in order to perform a certain task or computation. On the one hand, the same algorithm may be expressed in many different languages and still be the same algorithm even though it may look quite different. On the other hand, different algorithms can be used to perform the same task. A recipe is an example of an algorithm that describes how to prepare a specific dish or meal. Consider for example the following algorithm that describes how one makes muffins.

1. Prepare ingredients: 2 cups flour, 1 tsp baking powder. 5 tbsp milk powder,
   1 egg, tbsp olive oil, 1 cup water
2. Turn oven on to 250 C.
3. Mix wet ingredients (water,olive oil, egg)
4. Mix dry ingredients (milk powder,flour,baking powder)
5. Pour wet into dry.
6. *Option: Mix 1 cup of fruit(i.e. blueberries) into batter.*
7. Pour batter into muffin tray.
8. Bake in oven 20 minutes
9. *If finished(brown), GOTO Step 11*
10. *Bake for 1 more minute. GOTO Step 9*
11. Take out of Oven

This recipe explains the steps one follows to make muffins. It provides all the informaton that one needs to know to be successful. The new cook does not need to rediscover anything. Ignoring steps 6, 9, and 10 (in italics), the flow is seen to be linear – no decisions to make. Step 6 is a conditional or option – a decision needs to be made about whether to add berries to the muffin. Steps 9 and 10 indicate repetition: they need to be preformed a number of times until a condition is met. As a final note, notice that the order in which steps 3 and 4 are completed is not important – in fact, they could be done in parallel. In the following section we will introduce UML diagrams that allow us to illustrate diagramatically this algorithm.

3.2 UML Diagrams

Expressing your algorithm clearly before starting to write computer code is crucial for creating easy to understand, well structured code. In order to help you to do this, a standard, called Universal Modelling Language, UML for short, has been developed to help you learn to think before you start to code.

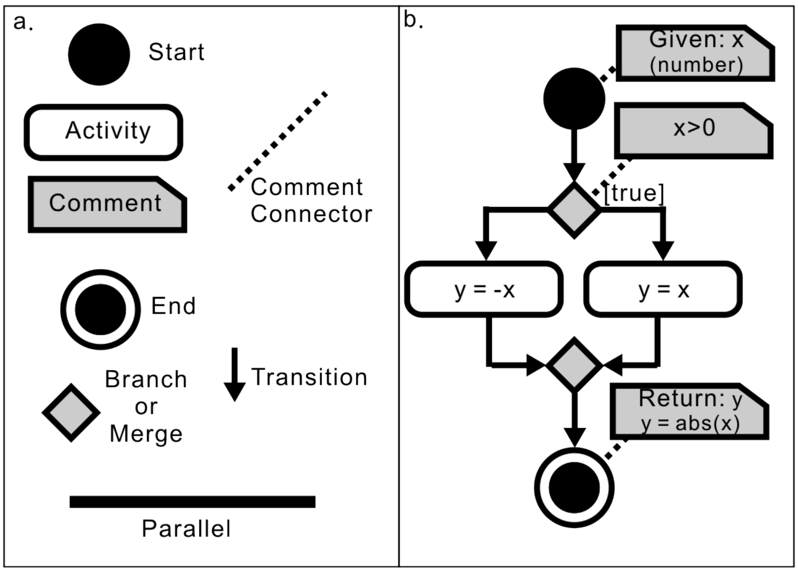


Fig. {UML} (a) Key symbols used to describe algorithms using UML activity diagrams. (b) Putting together the symbols to describe and algorithm to take the absolute value of a number.

Figure {UML} summarizes the key symbols that are used in UML activity diagrams. These diagrams are used to help us to show the steps in an algorithm. Fig. {UML} (a) identifies the symbols while Fig. {UML} (b) uses these symbols to describe the process are placed together to describe an algorithm to take the absolute value of a arbitrary number.

For our UML diagrams, we use a total of eight symbols: Start, End, Activity, Comment, Branch/Merge, Transition, Comment Connector, and finally Parallel. Within a given UML diagram, there should only be one start symbol. Activities are conducted in the order they appear in the diagrams. In the case that order is not important for two activities, then one can use the Parallel bar to indicate this. In Fig. *{UML}* (b), one can see that the algorithm starts with a number (x). Next a decision is made: if x>0 then we take the right path. If x<0 we take the left path. In the case that we take the right path, the we just store the value of x in the variable y. In the case that we took the left path, we change the sign of x and store the result in y. The paths then join back together again and the algorithm ends. Note that if we take the right path, then we do not take the right path.

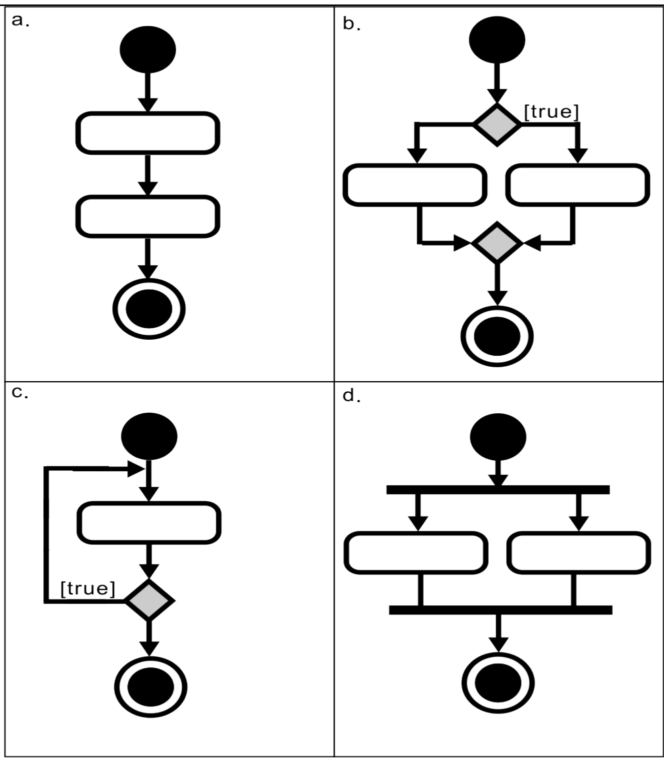


Fig *{UML-flow}* UML diagrams illustrating the four basic types of flow (a) linear-sequential (b) conditional, (c) repetition (d) parallel processing.

Figure *{UML-flow}* illustrates the four types of flow that we can implement in a program. All algorithms can be expressed in terms of a combination of these flow structures. In Fig.*{UML-flow}*(a) linear-sequential flow is illustrated. In this type of flow, the activity in the upper box is first completed and then activity in the next box can be started. In (b) conditional flow is illustrated. Based on the the value of a variable or some condition either the right path is taken or the left path is taken. Both paths are never taken. In (c) repetition is illustrated. In this type of flow, a given activity is repeated until some condition is met. Finally in (d) a specialized type of flow is illustrated: “parallel” processing. In this case, both activities are completed but the order in which they are completed is not important. Thus they can be done in parallel. In contrast to sequential processing, there is no order for activities. In contrast to conditional processing, both branches are taken.

4. Networks and Operating Systems

5. Number Systems and Data Handling

5.1 Representing Numbers

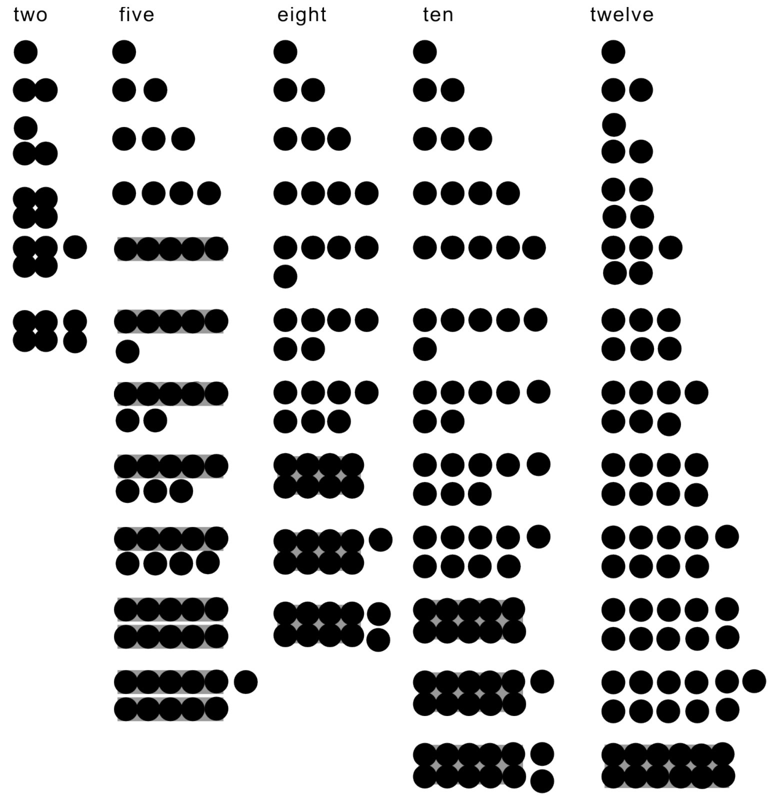


Fig. *{num-grp}* Ways of dividing numbers into groups

A number system can be thought of as a way to group, symbolize and express a number or quantity of objects to make counting and calculations involving these objects simpler. Fig. *{num-grp}* illustrates ways of grouping numbers: groups of two, groups of five, groups of eight, groups of 10, groups of 12. In England, groups of twelve are often used, being denoted as a dozen. So we then can then count five dozen and three eggs.

Our next step is to determine how to write and symbolize numbers. Table *{numsym}* gives some examples of symbols for various numbers.

Table *{numsym}* Glyphs to Represent Numbers

English	Chinese	Arabic	Roman	English	Chinese	Arabic	Roman
zero	零	0		eleven		B	
one	一	1	I	twelve		C	
two	二	2		thirteen		D	
three	三	3		fourteen		E	
four	四	4		fifteen		F	
five	五	5	V	sixteen		G	
six	六	6		seventeen		H	
seven	七	7		eighteen		I	
eight	八	8		nineteen		J	

English	Chinese	Arabic	Roman	English	Chinese	Arabic	Roman
nine	九	9		fifty			L
ten	十	A	X	hundred	白		C

Once one has decided how to group objects and symbolize numbers, the final question involves how to use the symbols to represent these numbers. There are two choices – a non-positional and positional number systems. In a non-positional number system a symbol has the same value wherever it is written. In a positional number system a symbol's value is determined by its location. For example, consider the symbol 'a' and the symbol 'b'. For a positional number system, the value of 'ab' is different from the value of 'ba'. (21 means something different from 12.) The actual meaning of 21 depends on how we have decided to group the numbers. If you have decided to think in terms of groups of ten than this means two groups of ten plus one . If you have decided to think in terms of groups of twelve, than this means two dozen plus one. Computers generally work in base 2. But for humans it is awkward to have to have a long string of 0s and 1s. Thus we usually try to think in terms of an intermediate base: either base 8 or base 16. Table {numrep} compares a number of ways to represent the numbers from zero to sixteen.

Table *{numrep}* Glyphs to Represent Numbers

English	中	Roman	Positional Systems (Base)			
			Base two	base eight	base ten	base sixteen
zero	零		0	0	0	0
one	一	I	1	1	1	1
two	二	II	10	2	2	2
three	三	III	11	3	3	3
four	四	IV	100	4	4	4
five	五	V	101	5	5	5
six	六	VI	110	6	6	6
seven	七	VII	111	7	7	7
eight	八	VIII	1000	10	8	8
nine	九	IX	1001	11	9	9
ten	十	X	1010	12	10	A
eleven		XI	1011	13	11	B
twelve		XII	1100	14	12	C
thirteen		XIII	1101	15	13	D
fourteen		XIV	1110	16	14	E
fifteen		XV	1111	17	15	F
sixteen		XVI	10000	100	16	10

As you learn to calculate in different bases, you will initially find it awkward and maybe a little difficult. This is not because different bases are more difficult but rather because you have memorized and become used to base 10. (If in primary school you had learned base 8, then this base would seem natural to you. For example, using base six, you could use your right hand's fingers for counting individuals and your left hand fingers to hold the number of groups of six!)

With this introduction, we can now look the definition of a positional number system more formally in a way that allows us to include fractions:

$$\begin{aligned}&(\ldots S_i \ldots S_1 S_0 . S_{-1} S_{-2} \ldots S_{-i} \ldots)_b \\ n = &\ldots + S_i \times b^i \ldots + S_1 \times b^1 + S_0 \times b^0 + S_{-1} \times b^{-1} + S_{-2} \times b^{-2} \ldots + S_{-i} \times b^{-i} \ldots\end{aligned} \qquad (1)$$

Consider now a number represented by the symbols n=**11.10**. (Lets assume that the number represents the number of cookies in a cookie jar. We can then express the number as:

$$n \; = \; 1 \times 10 \; + \; 1 \times 1 \; + \; \frac{1}{10} \; + \; \frac{0}{100} \qquad (2)$$

If *n* is in base ten, then we have eleven complete cookies and one tenth of a cookie. If *n* is in base two, then we have three and one half cookies in the cookie jar. If *n* is in base sixteen then we have seventeen and one sixteenth of a cookie in the cookie jar.

5.2 Converting between Bases in a Positional Number System

Since a computer works in base two and humans generally use base ten, we need to be able to convert between bases. While one can do the conversion directly, it is easiest to convert via base sixteen, i.e., $ten \longleftrightarrow sixteen \longleftrightarrow two$. The reason for using base sixteen is that the maximum number before grouping in this base is fifteen. Fifteen can be broken down into one group of eight, one group of four, one group of two and one group of one. Expressed numerically $F_{(sixteen)} = 1111_{(two)}$ Figure *{numgrp16}* provides a schematic illustrating the grouping of objects in base sixteen.

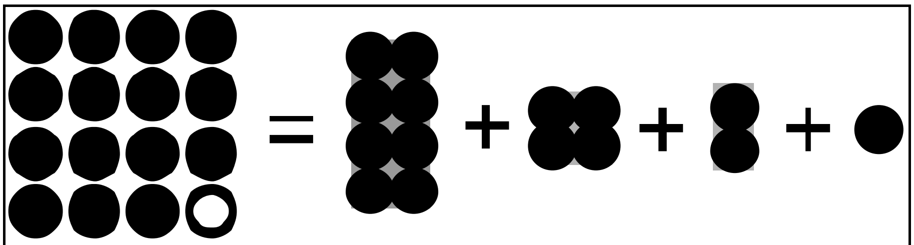


Fig. *{numgrp16}* Relating a group of objects in base sixteen to base two

We will summarize conversion with two short examples: the first for an integer and the second for a fraction. Table *{convertFixed}* gives the example for an integers while Table *{convertFraction}* gives an equivalent example for a fraction before summarizing the process with UML diagrams.

Table {convertFixed} Converting Between Bases for an Integer

Step	base ten	base sixteen	base two	comments
Input	**59**			
Convert to base sixteen	59/16 = 3 R 11	B		three groups of sixteen, eleven left over ($11_{10} = B_{16}$)
	3/16 = 0 R 3	3B		no groups of sixteen squared
Split		3 B		
Convert to base two		3 B	0011 1011	see Fig. *{numgrp16}*
Answer/ Input			**111011**	
Group			0011 1011	
Convert to base sixteen		3 B	0011 1011	see Fig. *{numgrp16}*
Convert to base ten	11	3B		($B_{16} = 11_{10}$)
	3 * 16 + 11	3		three groups of sixteen + eleven

Step	base ten	base sixteen	base two	comments
Answer	**59**			

Table ***{convertFraction}*** **Converting Between Bases for a Fraction**

Step	base ten	base sixteen	base two
Input	**0,59**		
Convert to base sixteen	0,59 x 16 = 9,44	0,9	
	0,44 x 16 = 7,04	0,97	
	0,04 x 16 = 0,64	0.970	
	0,64 x 16 = 10,24	0,970A	
Split		0 , 9 7 0 A	
Convert to base two		0 , 9 7 0 A	0000, 1001 0111 0000 1010
Answer/Input			**0,100101110000101**
Group			0000, 1001 0111 0000 1010
Convert to base sixteen		0 , 9 7 0 A	0000, 1001 0111 0000 1010
Convert to base ten	10/16 = 0.625	0,970A	
	(0 + 0.625)/16	0,970	
	(7 + 0.0390625)/16	0,97	
	(9 + 0.4399414063)/16	0,9	
Answer	0.5899963379		

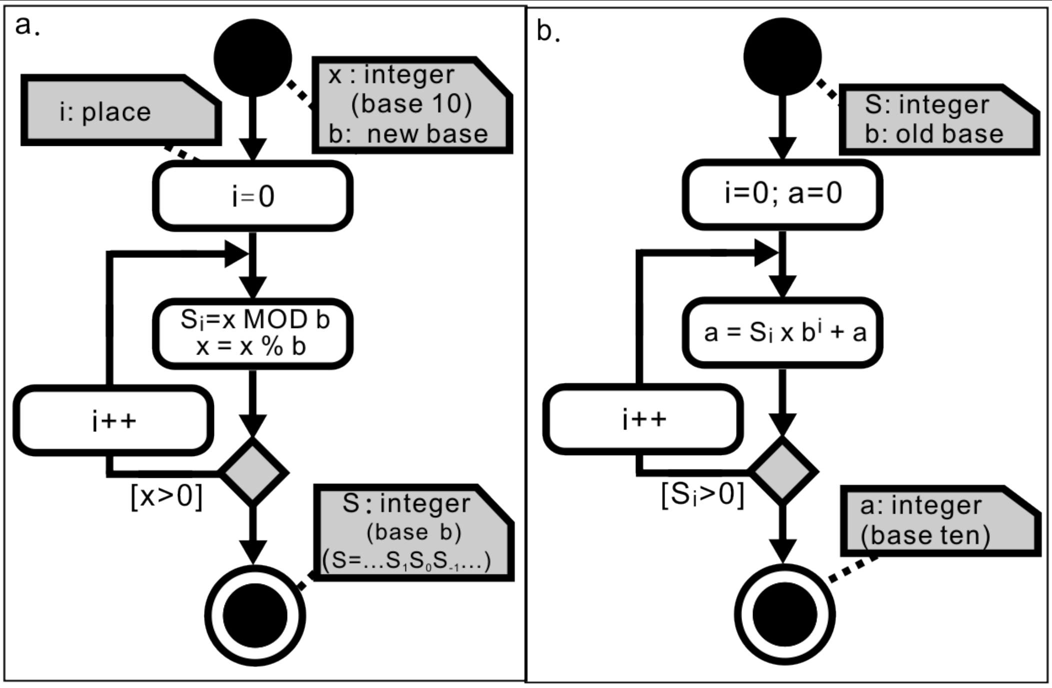


Fig. ***{convUMLint}*** **UML activity diagram illustrating the conversion of an integer between bases using calculations done in base ten. (a) Conversion from base ten to a new base (b) Conversion from an arbitrary base back to base ten.**

Note that in converting back and forth the final number is not exactly the same as the number we input – this is a common problem when using a computer to work with fractions. This problem has led to the development of two ways to represent number: fixed-point and floating-point.

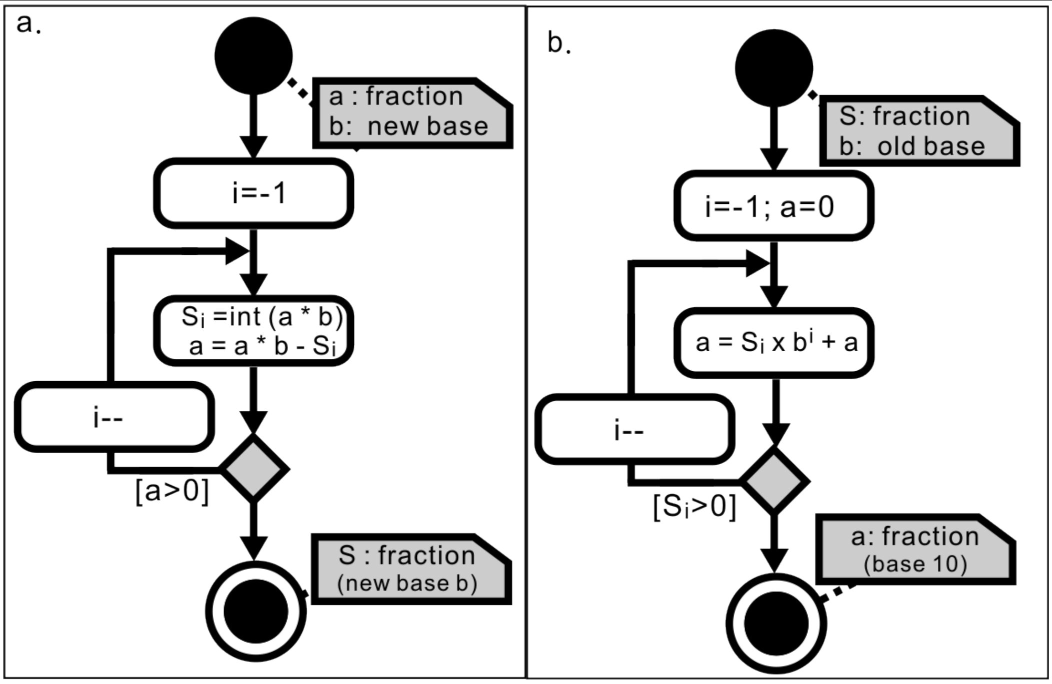


Fig. *{convUMLfrac}* **UML activity diagram illustrating the conversion of a fraction between bases using calculations done in base ten. (a) Conversion from base ten to a new base (b) Conversion from an arbitrary base back to base ten.**

The process of converting between bases can be expressed with UML diagrams as shown in Fig. *{convUMLint}* for the integer portion of the number and in Fig. *{convUMLfrac}* for the fraction part of the number. In these diagrams, i++, indicates the incrementation of i by the addition of 1, i-- indicated decrementing i by 1. MOD indicates taknig the remainder, while % is indicative of integer division. For example, assuming the x =11 and b=3

```
y = x MOD b = 11 MOD 3 = 2
y = x % b = 11 % 3  = 3
y = x / b =  11 / 3 = 3.66666...
y = int(x/b) = int(11/3) = 3
```

Finally note that the choice of directly converting from base ten to base two or using an intermediate base (i.e. base sixteen) is entirely up to you as a user.

5.3 Storage of Numbers

Once your data has been converted from base ten to base two, one needs to decide how to store the data. In modern computers there are generally two choices: fixed-point storage or floating -point storage. Both have their advantages and disadvantages.

Fig. *{umlstore}* shows a UML diagram indicating how an integer expressed in base two, is converted into either sign and magnitude or 2s complement format and stored in computer memory. Note that if the integer is positive, then there is no difference in the way the number is stored.

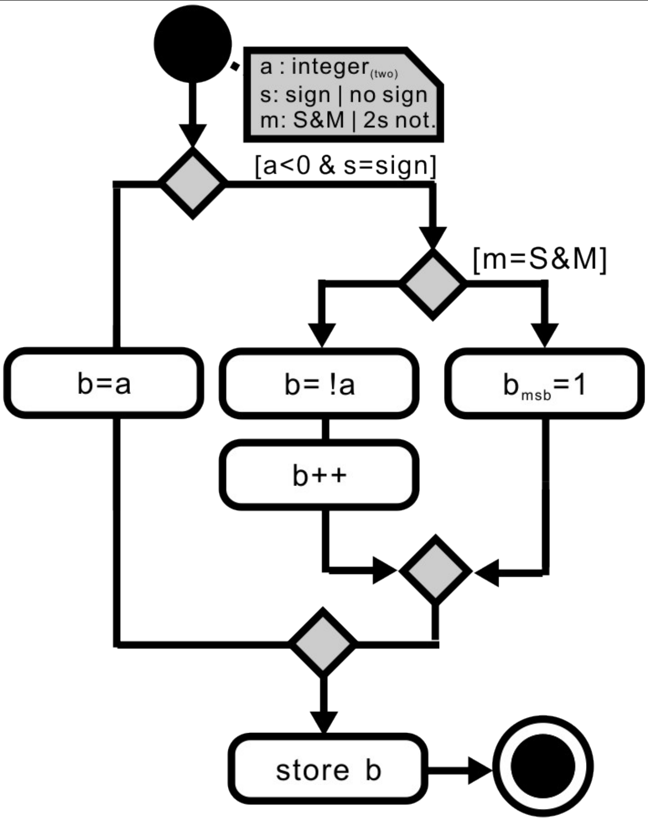


Fig. *{umlstore}* UML activity diagram illustrating the storage of an integer (in base two) in either sign and magnitude (S&M) or 2s Complement format (msb=most significant bit)

6. Computer Organization

7. Introduction to ANSI-C Programming

8. Tutorials

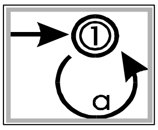
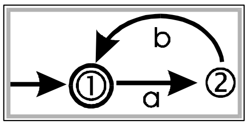
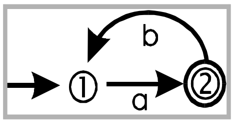
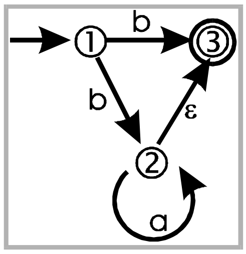
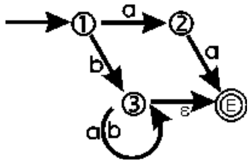
8.1 FSA (Finite State Automata) #1

J D White

GROUP: ________________

FSA 有限狀態自動機
RE 正則表達式
DFSA/NFSA 確定性/具有不確定性

a. What languages is accepted by each FSA? (Answer as a regular expression.)

 **a** ________	 **b**________	 **c** ____________	 **d** ____________	 **e** __________ __________

b. Which one of the above FSAs is deterministic (i.e. DFSA)?

____

c. Draw a DFSA for each regular expression. Convert to NFSA.

RE	Σ	DFSA	NFSA
a*b*c	**a,b,c**		
(a | b)*	**a,b**		
(a*|b*)	**a,b**		
(a*b)|(c+d)	**a,b,c,d**		

8.2 FSA (Finite State Automata) #2

J D White

GROUP: ________________

FSA	有限狀態自動機
RE	正則表達式
DFSA/NFSA	確定性/具有不確定性

a. Complete questions in Ian Chai, *Structuring Data and Building Algorithms*, pg 383-384.
Write your answers on this tutorial sheet. Be prepared to explain your solutions.

1. Answer

2. NFA's....

a	b	c

3. Give Regular Expressions.

a.
b.
c.

8.3 Turing Machine: Subtract 1

Ian Chai, J D White,

GROUP: ________________

binary number 二进制数
Turing Machine (TM) 圖靈機

a. Run Turing Machines

1. Download the TM Simulator from http://www.sdba.info/theory/turmachi.htm
2. Run the three machines with their respective tape. Try to follow what is happening. Modify the tapes and/or program and observe what happens.

b. Run this TM with given input tape and write final tape. Convert the Tuples representation of the controller to a state diagram diagram representation. q_o={MR}

TM Program (Tuples Representation)

```
;add1.tm:add 1 to binary number
(MR,0,0,>,MR) ; Move Right
(MR,1,1,>,MR)
(MR, , ,<,ADD)
(ADD,0,1,>,ML) ; ADD one
(ADD, ,1,>,ML)
(ADD,1,0,<,ADD)
(ML,0,0,<,ML) ; Move Left
(ML,1,1,<,ML)
(ML, , , ,STOP)
```

Tape (input → final)	State Diagram Representation
110 →	
1 →	
0 →	
111 →	

c. Write a TM control file (.tm) to subtract 1 from a binary number. Express your answer in both tuples representation (for computer) and state diagram (for humans) representations. Test run with TM Simulator.

TM Program (Tuples Representation)	Test Tape (input → final)
	1 → 0
	11 → 10
	10 → 01
	11100 → 11011
	TM Program (State Diagram Representation)

8.4 Turing Machine: Add 2

Ian Chai, J D White

GROUP: ____________________

binary number 二进制数
Turing Machine 圖靈機

a. Run Turing Machines

1. Download the TM Simulator from http://www.sdba.info/theory/turmachi.htm
2. Run the three machines with their respective tape. Try to follow what is happening. Modify the tapes and/or program and observe what happens.

b. Run this TM with given input tape and write final tape. Convert the Tuples representation of the controller to a state diagram diagram representation. q_o={MR}

TM Program (Tuples Representation)	Tape (input → final)	State Diagram Representation
;add1.tm:add 1 to binary number (MR,0,0,>,MR) ; Move Right (MR,1,1,>,MR) (MR, , ,<,ADD) (ADD,0,1,>,ML) ; ADD one (ADD, ,1,>,ML) (ADD,1,0,<,ADD) (ML,0,0,<,ML) ; Move Left (ML,1,1,<,ML) (ML, , , ,STOP)	101 → 1 → 0 → 111 →	

c. Write a TM control file (.tm) to add 2 to a binary number. Express your answer in both tuples representation (for computer) and state diagram (for humans) representations. Test run with TM Simulator.

TM Program (Tuples Representation)	Test Tape (input → final)
	0 → 10
	1 → 11
	100 → 110
	TM Program (State Diagram Representation)

8.5 Algorithms & UML

J D White

GROUP: ________________

a. Express in UML an algorithm that adds two numbers, input by the user, and prints the result.

b. Express in UML an algorithm that sums & averages six numbers together. Print the results.

c. Express in UML an algorithm that requests the user to input his sex and displays a message based on user's input.

For example, if the user is female, print: "You are very beautiful." Otherwise, the algorithm should print: "You are very handsome."

8.6 Number System Conversion

J D White

GROUP: ________________

Base 10 十进制数
Base 2 二进制数
Base 16 十六进制数

a. Convert From Base 10 to Base 2

Integer	Real
16 → **10 000**	2.1 →
21 →	5.3 →
95 →	11.75 →
132 →	21.3 →

b. Convert From Base 2 to Base 10

Integer	Real
1111 → **15**	0.1 →
1101 →	1.101 →
1 →	11.01 →
11110011 →	110.1 →

c. Convert From Base 2 to Base 16

Integer	Real
1111 → **15** → **F**	0.1 →
1101 → →	1.101 →
1 → →	11.01 →
11110011 → →	110.1 →

d. Base 16 to Base 2

Integer	Real
16 → **0001 0110**	0.1 →
21 →	1.101 →
95 →	11.01 →

e. Base 16 ⇔ Base 8

Sixteen to eight	Eight to Sixteen
16 →	16 →
21 →	21 →

8.7 Storing Data: Fixed and Floating-Point 1 byte (8 bits) 1 字节

J D White

GROUP: ________________

a. Unsigned Fixed Point Storage Using 8 bits (`unsigned char`)

Integer (Q8.0)	Real (Specify your Qm.n, not 1 answer)
$16_{(10)}$ → 1 0000 → **0001 0000**	$2.1_{(10)}$ → 10.00011 → **10000110 (Q2.6)**
$21_{(10)}$ → 1 0101 →	$5.3_{(10)}$ → 101.01001 →
$95_{(10)}$ → 101 1111 →	$0.75_{(10)}$ →
$132_{(10)}$ → 1000 0100 →	$21.3_{(10)}$ →

b. Signed Fixed Point Storage Using 8 bits (`char`)

Input	Sign & Magnitude	2s Complement
-16 →	**1001 0000**	**1111 0000**
+16 →		
-95 →		
-18 →		
+132 →		
-5.3 →	(Q7.0)	(Q7.0)
	(Q3.4)	(Q3.4)
-5.3 → **-53**		

c. Floating-Point Storage using IEEE Excess_127 format (32 bits) (`double`)

Input (10)	Scientific Notation (base 2)	IEEE Excess_127 format		
		S ±	Exp + 127 (8) Shifter	Mantissa (23) Fixed-Point Number
5.75	**101.11=1.0111x2**^{2}	**0**	**1000 0001**	**0111 0000 0000 00000000000**
17				0000 00000000000
-18				0000 00000000000
95				0000 00000000000
16				0000 00000000000
2.5				0000 00000000000
-21.3				0000 00000000000
0.75				0000 00000000000
0				0000 00000000000

8.8 Storing Data: Text

1 byte (8 bits) 1 字节
Glyph 雕文

J D White

GROUP: ________________

a. Store the following characters in 8-bits of memory (`unsigned char`). Use ASCII encoding.
(See Appendix or https://www.xiaotu.com/sdba/general/ascii.htm)

Glyph	**Code (Base 10 and Base 16)**	**Binary Storage**
A	(41_{16}=65_{10})	0100 0001
6		
Z		
!		
~		
*		

b. Store the following characters in 16-bits of memory (`unsigned long int`). Use UTF-8 encoding.
(See http://www.pinyin.info/tools/converter/chars2uninumbers.html)

Glyph	**Code (Base 16)**	**Binary Storage**
A	(0065_{10}= 0041_{16})	0000 0000 0100 0001
6		
Z		
我		
爱		
你		

8.9 Bit Operations: Logic & Shift

J D White

GROUP: ________________

a. LOGIC Operations: NOT(!), OR(||), AND(&&), XOR (`unsigned char`)

X	$X_{(16)}$	$X_{(2)}$	$X_{(16)}$	$X_{(2)}$	$X_{(16)}$	$X_{(2)}$	$X_{(16)}$	$X_{(2)}$
A	99		FF		00		01	
B	98		99		99		99	
!A								
A || B								
A && B								
A XOR B								

b. Logical Shift: LSHFT LEFT/RIGHT (`unsigned char`)

$X_{(10)}$	X_{16}	$X_{(2)}$	LSHFT-LEFT			LSHFT-RIGHT			Comment
			base 2	16	10	base 2	16	10	
89	59	0101 1001	1011 0010	B2	178	0010 1100	2C	44	Div. Truncate
3									
10									
32									
99									

c. Arithmetic Shift: ASHFT LEFT/RIGHT (`char`), 2s Complement Storage

$X_{(10)}$	X_{16}	$X_{(2)}$	ASHFT-LEFT			ASHFT-RIGHT			Comment
			base 2	16	10	base 2	16	10	
+3	3	0000 0011	0000 0110	6	6	0000 0001	1	1	DIV < 1/2
-3									
+10									
-10									
+89									

d. Circular Shift: CSHFT LEFT/RIGHT (`unsigned char`)

$X_{(10)}$	X_{16}	$X_{(2)}$	CSHFT-LEFT			CSHFT-RIGHT			Comment
			base 2	16	10	base 2	16	10	
3	3								
17									

8.10 Binary Arithmetic: Fixed-Point

J D White

GROUP: ________________

a. 2s Complement Fixed-Point (`char`)

X	$X_{(16)}$	$X_{(2)}$	$X_{(16)}$	$X_{(2)}$	$X_{(16)}$	$X_{(2)}$	$X_{(16)}$	$X_{(2)}$
A	+3	`0000 0011`	+3		+32		+32	
B	+2	`0000 0010`	-2		+09		-09	
A+B	+5	`0000 0101`						
$\overline{B}+1$	**-2**	`1111 1110`						
A-B	**+1**	`0000 0001`						

b. S & M Integers: Using 8 bits (Needed for Floating-Point Work)

X	$X_{(16)}$	$X_{(2)}$	$X_{(16)}$	$X_{(2)}$	$X_{(16)}$	$X_{(2)}$	$X_{(16)}$	$X_{(2)}$
A	+3	`0000 0011`	+2		+32		+7F	
B	+2	`0000 0010`	-3		+09		-7C	
A+B	**+5**	`0000 0101`						
A-B	**+1**	`0000 0001`						

8.11 Binary Arithmetic: Floating-Point

J D White

GROUP: ________________

a. Example (IEEE Excess_127)

	X		IEEE Excess_127			Denormalize....		
	base10	base 2	S	Exp (127+E)	Mantissa	S	Exp	Mantissa
A	+3.5	11.1=1.11e1	0	1000 0000	1100 00	0	1000 0001	1110 0000 00
B	+0.75	0.11=1.1e-1	0	0111 1110	1000 00	0	0111 1111	1100 0000 00
Align (small to larger radix)					(B)	0	1000 0001	0011 0000 00
A+B	+4.25	0100.01	0	1000 0001	0001 00	0	1000 0010	1000 1000 00

b. Two Positive

			IEEE Excess_127			Denormalize....		
X	**$X_{(10)}$**	**$X_{(2)}$**	**S**	**Exp**	**Mantissa**	**S**	**Exp**	**Mantissa**
A	+0.5							
B	+6.0							
Align (small to larger radix)								
A+B								

c. One Positive and One Negative

			IEEE Excess_127			Denormalize....		
X	**$X_{(10)}$**	**$X_{(2)}$**	**S**	**Exp**	**Mantissa**	**S**	**Exp**	**Mantissa**
A	+0.5							
B	-6.0							
Align (Smaller to Larger)								
A+B								

d. Check

			IEEE Excess_127			Denormalize....		
X	**$X_{(10)}$**	**$X_{(2)}$**	**S**	**Exp**	**Mantissa**	**S**	**Exp**	**Mantissa**
A	+10							
B	+01							
Align (Smaller to Larger)								
A+B								

8.12 Assembly Language Programming

J D White

GROUP: ________________

a. Write an Assembly languagecode to flip the sign of a (2s comp) fixed-point number

1. Get a number from the keyboard (**MFE**) to Memory **M1C** Start Code at **M07**
2. Switch the sign of the number (take the 2s complement) in **M1C**. Place the result in **M1D**
3. Write the code to write the number in **M1D** to the printer (**MFF**).
4. End (Stop) the program.

Address	Code	Op-code	Action	Program Section
M07				**a. Input**
M08				
M09				**b. Calculation**
M0A				
M0B				
M0C				
M0D				**c. Output**
M0E				
M0F				**d. Return**

b. Write code to convert a number in Sign & M to 2s complement notation.

Assume the 16-bit number is held at address **M1C** in S & M notation . Hint: Use JUMP.

Add	Code	Op-code	Action	Comments	
M07	**10EF**	**LOAD**	**R0 ← MEF**	**Enter 0**	**Create Flag for Negative Numbers (in S&M notation)**
M08	**A000**	**INC**	**R0 ← R0++**		
M09	**E010**	**CSHFT**	**R0**	**10000000 00000000**	
M0A	**111C**	**LOAD**	**R1 ← M1C**	**Load Number to change format**	
M0B					**check if negative**
M0C					
M0D					**Deal with negative numbers, take 2s complement of magnitude**
M0E					
M0F					
M10					
M11					**stop**

c. Write Code to load your programs into computer memory. (Put in ROM).
Use the ASS assembly language code. Assume **R0=0** at start-up

Address	Code	Op-code	Action	Explanation
M01	**11FE**	**LOAD**		**Location to put program line 1 (M07)**
M02				
M03				
M04				
M05				
M06				
M07				
M08				
M09				
M0A				

d. Alternate Solution to Question c

Add	Code	Op-code	Action	Comments	
M07	**10EF**	**LOAD**	**R0 ← MEF**	**Enter 0**	**Create Flag for Negative Numbers**
M08	**A000**	**INC**	**R0 ← R0++**		
M09	**E010**	**CSHFT**	**R0**	**10000000 00000000**	
M0A				**Load Number to change format**	
M0B					**check if negative**
M0C					
M0D					**Deal with negative numbers, take 2s complement of magnitude**
M0E					
M0F					
M10					
M11					**stop**

8.13 Algorithms: From Concept → UML → ASS

J D White

GROUP: ________________

Write an algorithm to (1) Multiply a fixed-point number by 2 (2) If the answer is larger than 60, then return the product. If the number is less than 60, return zero. 寫出一個演算法:(1)對一個整數*2，(2)如果結果數字大於 60，回傳此數字；如果數字小於 60，則回傳 0。 Example: If input=31 then the program algorithm will return 62. If input=5 then the program will find that 10<60 and so will return zero. 例子:如果輸入是 31，那結果會回傳 62；如果輸入是 5，因為計算結果小於 60，則結果會回傳 0。

a. Write your answer in Pseudo-Code.

Algorithm:	
Purpose:	
Pre:	
Post:	
Return:	

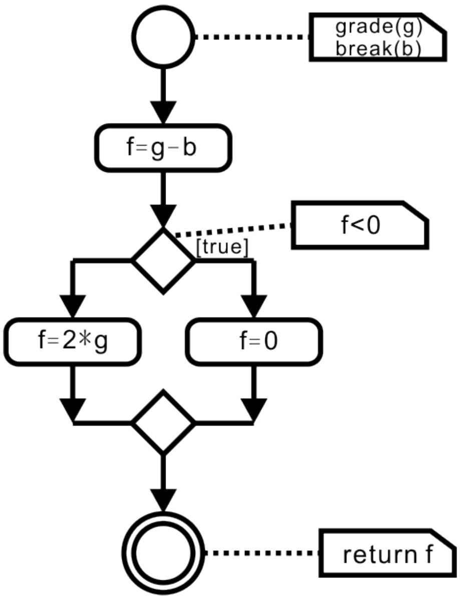
b. Write your answer as an UML activity diagram

c. Implement as code using ASS.

1. Assume that all registers have been initialized to 0 (zero).
2. Write code to get a number from the keyboard (&FE) to Memory. Start Code at &07$_{16)}$
3. Assuming the fixed-point data is held in Memory in &1C, write code to do the calculation. . Place the result in &1D. Start the Code at address &09$_{16)}$
4. Write the code to write the result to the printer (&FF).

Address	Code$_{(16)}$	Instruction	Action & Explanation
$07_{(16)}$			
$08_{(16)}$			
$09_{(16)}$			
$0A_{(16)}$			
$0B_{(16)}$			
$0C_{(16)}$			
$0D_{(16)}$			
$0E_{(16)}$			
$0F_{(16)}$			
$10_{(16)}$			
$11_{(16)}$			
$12_{(16)}$			
$13_{(16)}$			
$14_{(16)}$			
$15_{(16)}$			
$16_{(16)}$			
$17_{(16)}$			
$18_{(16)}$			
$19_{(16)}$			
$1A_{(16)}$			

d. Alternate solution with UML



Address	$Code_{(16)}$	Assembly	Explanation
$07_{(16)}$			
$08_{(16)}$			
$09_{(16)}$			
$0A_{(16)}$			
$0B_{(16)}$			
$0C_{(16)}$			
$0D_{(16)}$			
$0E_{(16)}$			
$0F_{(16)}$			
$10_{(16)}$			
$11_{(16)}$			
$12_{(16)}$			
$13_{(16)}$			
$14_{(16)}$			
$15_{(16)}$			
$16_{(16)}$			

*assumption that all registers initialized to zero is key to this program.

8.14 Simple ANSI-C Program

J D White

GROUP: ________________

a. Using jEdit, write a simple program (8 pnts: all or nothing)

1. Write a Simple Program

2. Save it

3. Compile it

4. Link it

5. Run it

b. Sample 2nd Questions Write a program to solve a problem (2 pnts)

Note: Teacher will choose which question you are assigned.

1. Simple Addition
   i. Declare a fixed-point and a floating-Point variable.
   ii. Initialize these two variables. (See http://www.xiaotu.com/tea/yzueo107/adatastr.swf)
   iii. Add the two variables together.
   iv. Print the result.
2. Subtraction
   i. Declare three fixed-Point variables.
   ii. Initialize the first two of these variables.
   iii. Subtract the 1st from the 2nd variables and store the result in the 3rd variable..
   iv. Print the values of all variables.
3. Multiplication (X)
   i. Declare three floating-Point variables.
   ii. Initialize the first two of these variables.
   iii. Multiply the 1st from the 2nd variables and store the result in the 3rd variable..
   iv. Print the values of all variables.
4. Division (/)
   i. Declare three floating-Point variables.
   ii. Initialize the first two of these variables.
   iii. Multiply the 1st from the 2nd variables and store the result in the 3rd variable..
   iv. Print the values of all variables.
5. Squaring (x^2)
   i. Declare one fixed-Point variables.
   ii. Initialize this variables.
   iii. Square the variable.
   iv. Print the result.

Assigned Problem:	
Solution Code	**Comments (Optional)**

9. Appendix

9.1 ASCII Encoding

Table A1: The ASCII encoding standard for text

Glyph	Dec	Hex	Glyph	Dec	Hex	Glyph	Dec	Hex
	32	20	@	64	40	`	96	60
!	33	21	A	65	41	a	97	61
"	34	22	B	66	42	b	98	62
#	35	23	C	67	43	c	99	63
$	36	24	D	68	44	d	100	64
%	37	25	E	69	45	e	101	65
&	38	26	F	70	46	f	102	66
'	39	27	G	71	47	g	103	67
(	40	28	H	72	48	h	104	68
)	41	29	I	73	49	i	105	69
*	42	2A	J	74	4A	j	106	6A
+	43	2B	K	75	4B	k	107	6B
,	44	2C	L	76	4C	l	108	6C
-	45	2D	M	77	4D	m	109	6D
.	46	2E	N	78	4E	n	110	6E
/	47	2F	O	79	4F	o	111	6F
0	48	30	P	80	50	p	112	70
1	49	31	Q	81	51	q	113	71
2	50	32	R	82	52	r	114	72
3	51	33	S	83	53	s	115	73
4	52	34	T	84	54	t	116	74
5	53	35	U	85	55	u	117	75
6	54	36	V	86	56	v	118	76
7	55	37	W	87	57	w	119	77
8	56	38	X	88	58	x	120	78
9	57	39	Y	89	59	y	121	79
:	58	3A	Z	90	5A	z	122	7A
;	59	3B	[	91	5B	{	123	7B
<	60	3C	\	92	5C	|	124	7C
=	61	3D	]	93	5D	}	125	7D
>	62	3E	^	94	5E	~	126	7E

?	63	3F	_	95	5F			

9.2 Key Words (Chinese-English Dictionary)

This table presents an index of key English terms along with their Chinese equivalent as used in this course.

Table B1: Chinese-English Dictionary of Key Words

中文(台灣)	English	Example
主記憶體	main memory	
暫存器	registers	
2 補數表示法	2s Complement Representation	
位址匯流排	address bus	
演算法	algorithm	
應用層	application layer	
算術邏輯單元	arithmetic logic unit;ALU	
人工智能	Artificial Intelligence; AI	
彙編語言	Assembly Language	
十進制數	Base 10	
十六進制數	Base 16	
二進制數	Base 2	
批次作業系統	batch operating systems	
二進制數	binary number	
匯流排拓樸	bus topology	
字節	byte	
快取記憶體	cache memory	
中央處理單元	central processing unit; CPU	
共用閘道介面	Common Gateway Interface；CGI	
控制匯流排	control bus	
資料匯流排	data bus	
資料鏈結層	data link layer	
死結	deadlock	
解碼	decode	
需求分頁	demand paging	
需求分段	demand segmentation	
設備管理者	device manager	
確定性/具有不確定性	DFSA/NFSA	
直接記憶體存取	direct memory access；DMA	

中文(台灣)	English	Example
分散式系統	distributed systems	
網域名稱	domain name	
網域名稱伺服器	domain name server；DNS	
乙太協定	Ethernet protocol	
超碼系統	Excess System	
執行	execute	
擷取	fetch	
檔案管理者	file manager	
檔案傳輸協定	File Transfer Protocol；FTP	
定點	fixed-point	6.2
浮點	floating-point	6.20E-023
框	frames	
有限狀態自動機	FSA	
圖形使用者介面	graphical user interface	
超文件標記語言	Hypertext Markup Language；HTML	
輸入／輸出子系統	input/output (I/O) subsystem	
輸入／輸出控制器	input/output controller	
指令暫存器	instruction register；IR	
整數	integer	9
網際網路郵件存取協定	Internet Mail Access Protocol；IMAP	
網際網路協定	Internet Protocol；IP	
中斷驅動 I/O	interrupt-driven I/O	
IP 位址	IP address	
工作	job	
鍵盤	keyboard	
機械碼	Machine Code	
機器週期	machine cycles	
記憶體管理者	memory manager	
記憶體對映 I/O	memory-mapped I/O	
網狀拓樸	mesh topology	
螢幕	monitor	
多重程式	multiprogramming	

中文(台灣)	English	Example
多功能網際網路郵件擴展	Multipurpose Internet Mail Extension；MIME	
網路	network	
網路層	network layer	
非儲存性設備	non-storage device	
運算碼	op-code	
與運算元	operand	。
作業系統	operating system	
頁	pages	
平行系統	parallel systems	
分割	partitioning	
效能	performance	
實體位址	physical addresses	
實體層	physical layer	
管線處理	pipe-lining	
埠號	port number	
網路郵局協定	Post Office Protocol；POP	
行程	process	
行程管理者	process manager	
程式	program	
程式計數器	program counter；PC	
程式化 I/O	programmed I/O	
虛擬程式碼	pseudo-code	
佇列	queues	
隨機存取記憶體	random access memory；RAM	
正則表達式	RE	
唯讀記憶體	read-only memory；ROM	
即時系統	real-time system	
可靠性	reliability	
環狀拓樸	ring topology	
選擇路徑	routing	
排程器	schedulers	
排程	scheduling	

中文(台灣)	English	Example
網路安全	security	
單一使用者作業系統	single-user operating systems	DOS
星狀拓樸	star topology	
飢餓	starvation	
狀態流程圖	state diagram	
儲存性設備	storage device	
串流控制傳輸協定	Stream Control Transmission Protocol；SCTP	
分時	time sharing	
傳輸控制通訊協定	Transmission Control Protocol；TCP	
傳輸層	transport layer	
圖靈機	Turing Machine	
統一塑模語言	Unified Modelling Language；UML	
通用資源定位器	Uniform Resource Locator；URL	
通用串列匯流排	Universal Serial Bus；USB	
使用者資料封包通訊協定	User Datagram Protocol；UDP	
使用者介面	user interface	
虛擬記憶體	virtual memory	

9.3 ASS Assembly Language

This is a summary of the instructions that one can use in doing the simple assembly language programming.

Table C1: Aray's Simple Assembly Language

Instruction	Code	Operands			Comments
HALT	**0**				
LOAD	**1**	**RD**	**MS**		RD ← MS Load data from memory S into register D
SAVE	**2**	**MD**		**RS**	MS ← RD Save data from register D into memory S
LADD	**3**	**RD**	**RA**		Load data to RD from memory address specified in RA
SADD	**4**	**RA**	**RS**		Save data from RS into memory address specified in RA
MOVE	**5**	**RD**	**RS**		RD ← RS
NOT	**6**				RD ← NOT (RS)
AND	**7**	**RD**	**RS1**	**RS2**	RD ← RS1 AND RS2 (bit-wise)
OR	**8**				RD ← RS1 OR RS2 (bit-wise)
XOR	**9**				RD ← RS1 XOR RS2 (bit-wise)
INC	**A**	**R**			R ← R + 1 (R++)
DEC	**B**	**R**			R ← R – 1 (R--)
ADD	**C**	**RD**	**RS1**	**RS2**	RD ← RS1 + RS2
JUMP	**D**	**R**	**PC=(&MD)**		if R!=R0 then GOTO specified line in program.
CSHFT	**E**	**R**	**n**		Circular Shift, n=0 Shift RIGHT else LEFT (no bit loss)
ASHFT	**F**	**R**	**n**		Arithmetic Shift, n=0 Shift RIGHT else LEFT

Table C2: Aray's Simple Assembly Language -- Examples

Machine Code	Assembly Code	Comment
`0000`	HALT	stop the program.
`1212`	LOAD R2 M12	copy data from memory location 12 to CPU register 2
`6110`	NOT R1 R1	flip bits of number in CPU register 1 and store result in register 1
`D212`	JUMP R1 PC=31	If CPU Register 1 is not equal to zero then set the program counter to get next instruction from memory location 31
`A100`	INC R1	Add one to the value in CPU Register 1

9.4 EO109 (Computer Programming) – The Follow-Up Course

EO109 is the followup course to EO107. In this course, we will first review the formation of a solution using using UML diagrams for a few problems and the assembly of a short piece of ANSI-C code. After that we will go on to study programming in depth as we work to convert out UML diagrams into ANSI-C code.

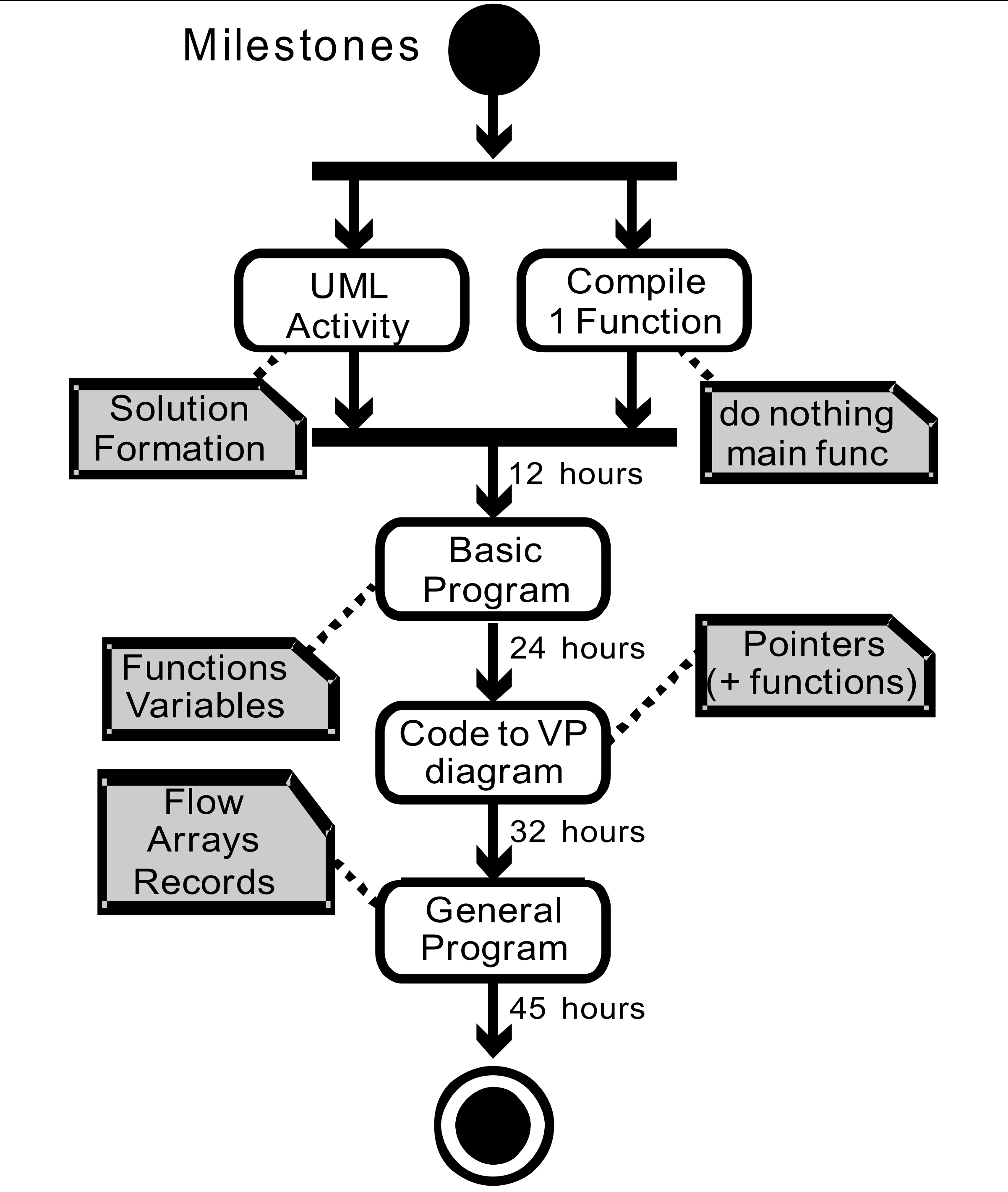


Fig. D1 UML diagram of the key milestones and content of EO109 Computer Programming

9.5 Group Member List

a. Up to 4 per group.
b. Work together on tutorials
c. One member may be called at random to represent the group. The group's mark depends on his/her performance
d. Leader receives bonus marks if group does well.

Table E1: Group Members

	Group Name:			Group Number:		
	Role	名字	Name	Student ID	Email	Hand-Phone
1	Leader*: 領導					
2	Member					
3	Member					
4	Member					

* Select one member as the group leader. He will be responsible for the work of the group

Table E2: Group Member Progress Form

	Name		Milestones						Participation	
	English	中文	1	2	3	4	5	6	c1	c2
1										
2										
3										
4										

Table E3: Attendance Record After Semester Midpoint

#	Class Number [base ten (base twelve)]											
	1	2	3	4	5	6	7	8	9	10(A)	11(B)	12(10)
1												
2												
3												
4												

Table E4: Attendance Record After Semester Midpoint

#	Class Number[base ten (base twelve)]											
	13(11)	14(12)	15(13)	16(14)	17(15)	18(16)	19(17)	20(18)	21(19)	22(1A)	23(1B)	24(20)
1												
2												
3												
4												

9.6 Example Tests for Milestones

a. Theory of Computing (M1)

J D White

GROUP:________

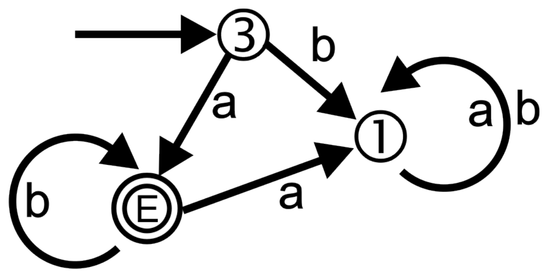
Name 名字: Sample Test w/Solution

ID 號碼: ________

課號碼________

1. (a) Draw the state diagram for a DFSA, having input alphabet Σ={a,b} that accepts the regular expression: **ab*** (b) Define the machine using a transition function T along with Q, q_o and F

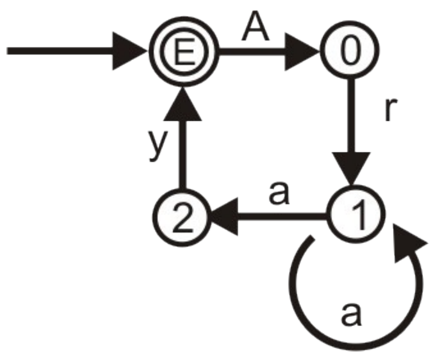
(a) Answer



(b) Answer

```
Q={1,3,E},qo={3},F={E},S={a,b}
T specified as...
[1,a,1]
[1,b,1]
[3,a,E]
[3,b,1]
[E,a,1]
[E,b,E]
```

2. (a) Express the FSA using transition functions (b) Express as a regular expression the string accepted by the NFSA shown below:



(a) **Q={E,0,1,2} qo={E}, F={E} T=**

```
[E,A,0]
[0,r,1]
[1,a,2];note-make deterministic
[2,a,2];note-make deterministic
[2,y,E]
```

(b) Answer: **(Ara+y)***

3. Given the following Turing Machine Controller:

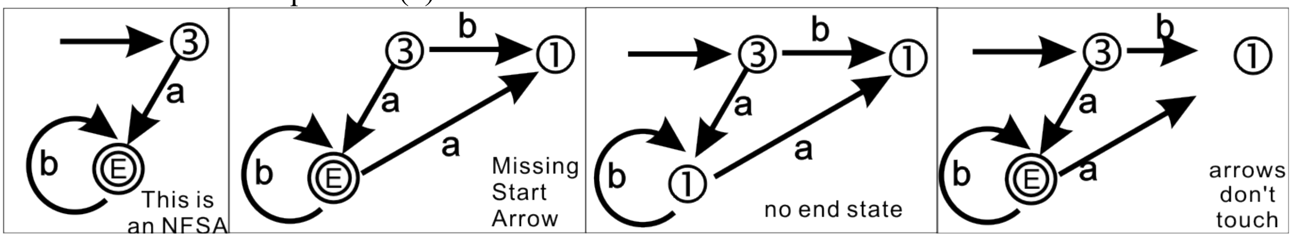
```
[b,' ', 0 ,R,c] ; b is initial state
[c,' ',' ',R,e]
[e,' ', 1 ,R,f]
[f,' ',' ',R,b]
```

Assuming the machine starts at the leftmost square of the following blank tape in state **b**, what does this machine write on the tape?

0		1		0		1		0

End of Test

Incorrect answers for question (1) with reasons....



b. Algorithms & UML (M2)

J D White

GROUP:________

Name 名字: Sample Test w/Solution

ID 號碼: ________

課號碼________

1. Provide an informal definition of the word "Algorithm". (演算法的定義:)

按部就班的解一個問題或完成某項工作的方法。**A step by step method for solving a problem or doing a task**

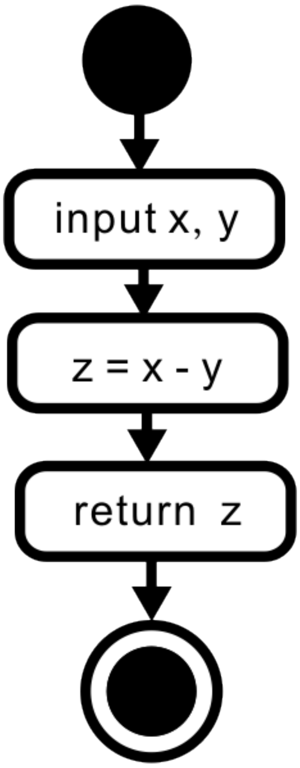
2. Determine the relationship between two positive integers. IF the integers are equal, you should return 0. IF the first is bigger then you should return a positive number, if the second is bigger, you should return a negative numbers-fixed

Example 1: if the user inputs 5 and 5 your algorithm should return the value 0 (例如,若进入爲 5 和 5，則 output 爲 0)

Example 2: If the user inputs 6 and 10 your algorithm should return -4. (例如,如果进入爲 6 和 10，則 output 爲-4)

Pseudo-code	
Algorithm:	equal(first,second)
Purpose:	To check to see if two integers are equal
Pre:	Given: two positive integers (first and second)
Post:	None
Return:	integer (res):
Steps:	res = first - second
	return(res)

UML Activity Diagram



c. Networks and Operating Systems (M3)

J D White

GROUP:________

Name 名字: Sample Test w/Solution

ID 號碼: ________

課號碼________

1. Name the 5 layers in the TCP/IP protocol.(請寫出 TCP/IP 協定的五個階層:)

Application 應用層	**Transport** 傳輸層	**Network** 網路層	**data link** 資料鏈結層	**Physical** 實體層

2. A typical operating system (作業系統) has 5 key components. What are they?(一個典型的作業系統有五個主要的組成單元，請寫出:)

UI 使用者介面	**Memory Manager** 記憶體管理	**Process Manager** 行程管理者	**Device Manager** 設備管理者	**File Manager** 檔案管理者

d. Number Systems, Data Storage(M4)

J D White

GROUP:________

Name 名字: Sample Test w/Solution
ID 號碼: ________
課號碼________

1. Convert from Base 10 to Base 16 to Base 2

Base Ten	Base Sixteen	Base Two
7	7	111
-25.25	-19.4	-11001.01

2. Write the bit representation of these numbers (Base two) and characters in memory:

Input	Fixed (char)	Floating-Point (float Excess-127)
11	0000 0011	0 1000 0000 1000 0000 0000 0000 0000 000
-11	1111 1101	1 1000 0000 1000 0000 0000 0000 0000 000
1.1	Not required	0 0111 1111 1000 0000 0000 0000 0000 000
'Q'	0101 0001	Note use ASCII encoding. Not stored in floating point container

3. Logic Operations: Specify the logic operator (e.g. AND) and bit pattern (e.g. 0110) to change "in" to "out" . Use each logic operation only once.

IN	0	1	1	1		IN	0	1	1	1		IN	0	1	1	1
OR	1	0	0	0		AND	0	0	0	1		XOR	1	1	1	1
OUT	1	1	1	1		OUT	0	0	0	1		OUT	1	0	0	0

4. Addition and Subtraction (fixed-point storage) (short int) 16 bits

j	0000 0000 0000 0111	j+k	1111 1111 1110 0000
k	1111 1111 1101 1001	j-k	0000 0000 0010 1000

5. Addition and Subtraction (floating-point storage IEEE excess-127) (float)

x	Comments Explanation	0 1000 0001 1100 0000 0000 0000 0000 000
y		0 1000 0010 1010 0000 0000 0000 0000 000
x'	Denormalize x	0 1000 0010 1110 0000 0000 0000 0000 000
y'	Denormalize y	0 1000 0011 1101 0000 0000 0000 0000 000
x″	AlignRadix:Shift x'	0 1000 0011 0111 0000 0000 0000 0000 000
A	Add: y' + x″	0 1000 0011 10100 0000 0000 0000 0000 000
x+y	Normalize A	0 1000 0011 0100 0000 0000 0000 0000 000
y″	2s comp: NOT(y')+1	0 1000 0011 0011 0000 0000 0000 0000 000
R	R = x' + y″	0 1000 0011 1010 0000 0000 0000 0000 000
B	ovr=0:NOT(R)+1 sgn	1 1000 0011 0110 0000 0000 0000 0000 000
x-y	Normalize B	1 1000 0001 1000 0000 0000 0000 0000 000

6. Retrieve the values from memory and write in base two

Type	Bit Sequence	Base Two
Fixed (`short int`)	**1111 1111 1101 1001**	-100111
Float (excess 127)	**0 1000 0001 1100 0000 0000 0000 0000 000**	111

7. Convert from base two to base sixteen and base ten

Base Two	Base Sixteen	Base Ten
-1011	-B	-11
110.11	6.C	6.75

e. Computer Organization 計算機組織 (M5)

J D White

GROUP:________

Name 名字: Sample Test w/Solution
ID 號碼: ________
課號碼________

1. Name the 3 subsystems in a modern computer(寫出現代計算機中的三個主要子系統):

Central Processing Unit (CPU) 中央處理單元	Memory 記憶體	Input/Output 輸入 / 輸出

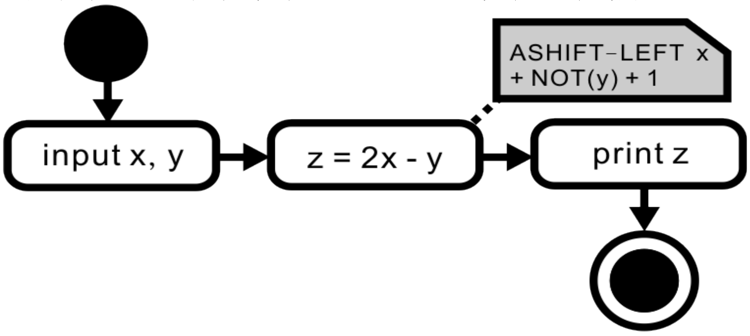
2. Name the 3 parts that make up the CPU(寫出組成 CPU 的三個主要部分):

Arithmetic logic unit (ALU) 邏輯運算單元	**Registers** 暫存器	Control Unit 控制單元

3. In program execution, a machine cycle includes 3 phases. What are they?(在程式執行中，一個機器週期包含三個部分，請寫出:)

Fetch 擷取	Decode 解碼	**Execute** 執行

4. Write a program in the ASS assembly language to read two numbers (x,y) from the keyboard , double the first and then subtract the second number. Print the result. You need to provide both the UML activity diagram and the ASS assembly language code. Assume that the keyboard is at &EF and the Printer is at &FF and that at the start all registers are initialized to zero（假設記憶體位置&EF 位置代表鍵盤，&FF 代表印表機）



Address	Code	Instruct	Action/Operands	Explanation
$07_{(16)}$	11EF	LOAD	R1 <-- MEF	Get the 1st number
$08_{(16)}$	12EF	LOAD	R2 <-- MEF	Get the 2nd number
$09_{(16)}$	F110	ASHFT	R1 Shift Left 1	Double the first number
$0A_{(16)}$	6220	NOT	!R2	Negate the 2nd number
$0B_{(16)}$	A200	INC		
$0C_{(16)}$	C312	ADD	R3 = R1 + R2	Add the doubled and negated.
$0D_{(16)}$	2FF3	SAVE	MFF <--R3	Print the result (equal)
$0E_{(16)}$				
$0F_{(16)}$				
$10_{(16)}$				

f. First Program in ANSI-C (M6)

J D White

GROUP:________

Name 名字: Sample Test w/Solution
ID 號碼: ________
課號碼________

1. Write a Simple Program in jEdit text editor.

```
int main(void){
  return(0);
}
```

2. Save it as an ASCII text file.

```
CTRL-S  c:\sdba\yyy.c
```

3. Compile it into a binary file.

```
gcc -c -ansi -Wall yyy.c
```

4. Link it with other binary files (if required) to make an executable program.

```
gcc -o yyy.exe yyy.o
```

5. Run it

```
yyy.exe
```