บรรณานุกรม

ภาษาไทย

ดารารัตน์ แซ่ลี้, "การพัฒนาตัวจำลองอิงกรรมวิธีสำหรับการประมวลผลแบบขนาน," การประชุมวิชาการสถิติประยุกต์ ครั้งที่ 8, หน้า 262 - 277, คณะสถิติประยุกต์ สถาบันบัณฑิตพัฒนบริหารศาสตร์, 2533.

ทักษิณา สวนานนท์, พจนานุกรมศัพท์คอมพิวเตอร์, บริษัทไฮเทคพริ้นติ้งจำกัด, กรุงเทพมหานคร, 2533.

ยืน ภู่วรวรรณ, "RISC and SPARC," ไมโครคอมพิวเตอร์, 78, 228-238, 2535.

ราชบัณฑิตยสถาน, ศัพท์คอมพิวเตอร์ฉบับราชบัณฑิตยสถาน, บริษัทเพื่อนพิมพ์จำกัด, กรุงเทพมหานคร, 2533.

ศิริวรรณ ฉันทาดิศัย, หลักการเขียนโปรแกรมภาษาแอสเซมบลี 8088, สำนักพิมพ์ประกายพรึก, กรุงเทพมหานคร, พิมพ์ครั้งที่ 2, 2534.

ภาษาอังกฤษ

Ferrari, Domenico, Computer System Performance Evaluation, Prentice-Hall Inc., 1978.

Hayes, John P., Computer Architecture and Organization, McGraw-Hill Inc., 1979.

Hwang K., and F.A. Briggs, Computer Architecture and Parallel Processing, McGraw-Hill Inc., 1984.

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

Kreutzer, Wolfgang, System Simulation Programming Styles and Languages, McGraw-Hill Inc., 1986.

Law, Averill M., and W. David Kelton, Simulation Modeling and Analysis, McGraw-Hill Inc., 1982.

Perry, Douglas L., VHDL, McGraw-Hill Inc., 1991.

Porter, Kent, Stretching Turbo C, Brandy, New York, 1989.

Rector, Russell, and George Alexy, The 8086 Book, McGraw-Hill, 1980.

Schwetman H., CSIM Reference Manual (Revision 9), Microelectronics and Computer Technology Corporation, 1986.

Smith, A. J., "Cache Memories," Computer Surveys, 14(3), 473-530, 1982.

Stallings, William Ph.D., Computer Organization and Architecture, Macmillan Publishing Company, 1987.

Uffenbeck, John, The 8086/8088 Family : Design , Programming and Interfacing, Prentice-Hall Inc., 1982.

Watcharawittayakul, W., "Managing Large Register Files, Cache memories, Instruction Buffers, and Write Buffers: A Reduced Instruction Set Computer Performance Study," Ph. D. Thesis, Univ. of Illinois at Urbana-Champaign, 1988.

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

______________________, "N-POST : A Process Oriented Simulation Tool," Final Report Submitted to the Research Promotion Committee, National Institute of Development Administration, Bangkok, 1989.

Wiecek, Cheryl A., "The Simulation of Processor Performance for the VAX 8800 family," Digital Technical Journal, 4(2), 100-110, 1987.

Weicker, R. P., "Dhrystone : A Synthetic Systems Programming Benchmark," Communications of the ACM, 27(10), 1013-1030, 1984.

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก ก

คู่มือการใช้งานตัวจำลองอิงกรรมวิธี np

1. การใช้ตัวจำลองอิงกรรมวิธี np

ในการเขียนโปรแกรมเพื่อทำการจำลองปัญหา ผู้ใช้ต้องมีแฟ้มข้อมูลชื่อ np.h ซึ่งแสดงถึงโครงสร้างข้อมูลและฟังก์ชันพิเศษที่ได้จัดไว้ให้เป็นส่วนเพิ่มเติมสำหรับการเขียนโปรแกรม ดังนั้นในโปรแกรมต้องมีประโยคคำสั่ง **#include "np.h"** เป็นจุดเริ่มต้นของโปรแกรม

1.1 โครงสร้างข้อมูล

โครงสร้างข้อมูลใน np.h กำหนดขึ้นเพื่อให้เหมาะกับลักษณะการทำงานของตัวจำลองอิงกรรมวิธี ได้แก่

PROCESS	เป็นข้อมูลของแต่ละกรรมวิธีที่กำหนดขึ้นในระบบ
FACILITY	เป็นข้อมูลของทรัพยากรและการเข้าใช้ทรัพยากรโดยกรรมวิธี
EVENT	เป็นข้อมูลของเหตุการณ์เพื่อให้กรรมวิธีต่างๆดำเนินการอย่างได้จังหวะกัน
MBOX	เป็นข้อมูลของการส่งข่าวสารเพื่อการติดต่อระหว่างกรรมวิธี
message	เป็นข้อมูลของข่าวสารที่ส่งผ่านให้กรรมวิธี
TABLE	เป็นข้อมูลทางสถิติที่ใช้ในการรายงานผลการทำงานของระบบ
QTABLE	เป็นข้อมูลทางสถิติที่ใช้ในการรายงานผลเกี่ยวกับแถวคอย
HISTOGRAM	เป็นข้อมูลที่บันทึกเพื่อรายงานผล
QHISTOGRAM	เป็นข้อมูลเกี่ยวกับแถวคอยที่บันทึกเพื่อรายงานผล

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

และการกำหนดตัวแปรในโปรแกรมที่มีโครงสร้างข้อมูลแบบใดแบบหนึ่งนี้ ทำได้ในลักษณะเดียวกันกับการกำหนดตัวแปรในภาษา C ตัวอย่างการกำหนดตัวแปรซึ่งชี้ไปที่โครงสร้างข้อมูลชนิด FACILITY กำหนดได้ดังนี้

```
FACILITY   *f;
```

1.2 <u>ฟังก์ชันพิเศษ</u>

ฟังก์ชันพิเศษที่พัฒนาขึ้นเพื่อใช้ในการเขียนโปรแกรมจำลองระบบ ผู้ใช้เรียกใช้ในโปรแกรมได้ในลักษณะเดียวกันกับฟังก์ชันมาตรฐานในคลังชุดคำสั่งมาตรฐานภาษา C เราสามารถแยกประเภทของฟังก์ชันเหล่านี้ได้ 4 ประเภทคือ

1.2.1 การกำหนด (Definition)

เมื่อเรียกใช้ฟังก์ชันประเภทนี้ ระบบจะกำหนดเนื้อที่ให้กับตัวแปรตามโครงสร้างข้อมูลของตัวแปรนั้น พร้อมกับกำหนดหน้าที่ของตัวแปรในโปรแกรมไปด้วย

ฟังก์ชันเหล่านี้ ได้แก่ gen(), facility(), nfacility(), event(), event_array(), mbox(), mbox_array(), table(), qtable(), histogram(), qhistogram()

1.2.2 การควบคุม (Control)

การจัดการให้การทำงานของแบบจำลองที่สร้างขึ้นนี้เป็นไปตามกระบวนการของระบบ ต้องเรียกใช้ฟังก์ชันประเภทนี้ ได้แก่ hold(), attach(), detach(), use(), state(), set(), reset(), wait(), wait_blk(), mbox_state(), receive(), send(), rm_facility(), rm_event(), rm_mbox() และ rm_mbox_array()

1.2.3 การเก็บรวบรวมข้อมูล (Data Collection)

เมื่อเรียกใช้ฟังก์ชันประเภทนี้ ระบบจะทำการบันทึกข้อมูลต่างๆเพื่อการรายงานผลทางสถิติหลังจากเสร็จสิ้นการทำงานของโปรแกรมได้แก่ ฟังก์ชัน record(), in_qtable() และ out_qtable()

1.2.4 การสร้างตัวเลขสุ่ม (Random Numbers)

ฟังก์ชันประเภทนี้เรียกใช้เพื่อสร้างตัวเลขสุ่มที่มีการกระจายของข้อมูลแบบต่างๆ ได้แก่ฟังก์ชัน expdev() และ prob()

1.3 <u>การกำหนดกรรมวิธี</u>

กรรมวิธีที่กำหนดขึ้นในแบบจำลองนั้น ต้องมีประโยคคำสั่งแรกของแต่ละกรรมวิธี เป็น gen(); ซึ่งเป็นการสร้างเนื้อที่และสิ่งแวดล้อมที่จำเป็น สำหรับการทำงานของกรรมวิธีนั้นๆ

ทั้งนี้กรรมวิธีแรกในโปรแกรม ต้องเป็น np() เท่านั้น ไม่มีฟังก์ชัน main() เช่นโปรแกรมภาษา C ทั่วๆไป ส่วนกรรมวิธีอื่น ผู้ใช้กำหนดชื่อกรรมวิธีได้ตามความต้องการ และการกำหนดตัวแปรในโปรแกรม ก็ใช้การพิจารณาเช่นเดียวกับการเขียนโปรแกรมตามปกติ เช่น ถ้าต้องการให้ทุกกรรมวิธีรู้จักตัวแปรใด ก็ต้องกำหนดให้ตัวแปรนั้นเป็นแบบส่วนกลาง

การเขียนโปรแกรมและการกำหนดกรรมวิธี ผู้ใช้ศึกษาได้จากตัวอย่างโปรแกรมซึ่งแสดงไว้ในส่วนท้ายของภาคผนวกนี้

1.4 <u>การติดตั้งและใช้งาน</u>

เมื่อสร้างโปรแกรมสำหรับการจำลองแบบปัญหาแล้ว ต้องส่งไปให้ตัวแปลภาษา C (C compiler) ทำการแปลโปรแกรมก่อน แล้วจึงเชื่อมโยง (link) object file(s) ของผู้ใช้ เข้ากับ np โดยใช้คำสั่ง

cc np.a ex.o

โดยที่ ex.o เป็น object file ที่ได้จากการแปลโปรแกรมของผู้ใช้ด้วยตัวแปลภาษา C และ np.a เป็นแฟ้มข้อมูลที่เก็บรวบรวมฟังก์ชันต่างๆสำหรับการเรียกใช้ในโปรแกรม

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

ในระบบเครื่องคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการยูนิกซ์ (UNIX Operating System) ตัวแปลภาษา C จะสร้างแฟ้มสำหรับปฏิบัติการให้ โดยมีชื่อว่า a.out ดังนั้น ผู้ใช้ทำการปฏิบัติการด้วยคำสั่ง

a.out <-s (number)> <-f (name)>

สำหรับตัวเลือก (option) ที่ผู้ใช้สามารถใช้ได้ คือ

-s (Stack Size in bytes) เป็นการกำหนดขนาดของสแตก (Stack) มีหน่วยเป็นไบต์ สำหรับแต่ละกรรมวิธีที่เกิดขึ้นในระบบ ถ้าไม่ระบุ ระบบจะกำหนดให้เป็น 6144 ไบต์

-f (Output Filename) ใช้กำหนดชื่อแฟ้มข้อมูลสำหรับเก็บข้อมูลของผลลัพธ์ที่ได้จากการปฏิบัติการโปรแกรม ถ้าไม่ระบุ ผลลัพธ์จากการปฏิบัติการนั้น จะปรากฏบนจอภาพ ซึ่งเป็น แฟ้มข้อมูลส่งออกมาตรฐาน (Standard Output File) ของระบบ

2. ประโยคคำสั่งของตัวจำลองกรรมวิธี np

ประโยคคำสั่งต่อไปนี้ ใช้เพื่อทำการจำลองแบบปัญหาโดยเฉพาะ โดยพัฒนาในรูปแบบของฟังก์ชันซึ่งเรียกใช้ได้ในโปรแกรมภาษา C

2.1 ประโยคคำสั่งสำหรับการกำหนด

2.1.1 gen()

เป็นคำสั่งสำหรับสร้างกรรมวิธีใหม่ในระบบ และจะต้องเป็นคำสั่งแรกของกรรมวิธีนั้นเสมอ

2.1.2 f = facility("name")

ใช้สำหรับกำหนดทรัพยากร f ที่มีชื่อ name ขึ้นมาในระบบ เพื่อให้กรรมวิธีเข้าดำเนินการตามระบบที่จำลอง โดยที่ทรัพยากรนี้มีจำนวนผู้ให้บริการเพียง 1 จุด

ค่าตัวเลขจำนวนเต็มที่ได้จากการเรียกใช้ฟังก์ชันนี้ จะใช้สำหรับการอ้างถึงทรัพยากรนั้น ในภายหลังได้

2.1.3 f = nfacility("name",n)

ใช้ในการกำหนดเช่นเดียวกับ facility("name") แต่ทรัพยากรนี้สามารถกำหนดให้มีจำนวนผู้ให้บริการ 1 จุดหรือ มากกว่า ก็ได้

2.1.4 ev = event("name")

ใช้สำหรับกำหนดเหตุการณ์ (event) ที่มีชื่อ name ขึ้นมาในระบบ เพื่อทำให้การติดต่อระหว่างกรรมวิธีได้จังหวะกัน

2.1.5 ev = event_array("name",n)

ใช้ในการกำหนดเช่นเดียวกับ event() แต่ระบบจะกำหนดเนื้อที่ให้ในลักษณะแถวลำดับ ซึ่งมีขนาด n เมื่อต้องการใช้งาน ผู้ใช้สามารถอ้างถึงแต่ละเหตุการณ์ได้ตั้งแต่ 0 ถึง n-1

2.1.6 mb = mbox()

ใช้ในการกำหนดเนื้อที่สำหรับตู้จดหมาย เพื่อเก็บข่าวสารพร้อมรายละเอียดที่จำเป็น

2.1.7 mb = mbox_array(n)

ใช้ในการกำหนดเช่นเดียวกับ mbox() แต่ระบบจะกำหนดเนื้อที่ให้ในลักษณะแถวลำดับซึ่งมีขนาด n เมื่อต้องการใช้งาน ผู้ใช้สามารถอ้างถึงตู้จดหมายแต่ละตู้ ได้ตั้งแต่ 0 ถึง n-1

2.1.8 tbl = table("name")

ใช้สำหรับกำหนดตาราง (table) ชื่อ "name" ขึ้นมาในระบบ เพื่อใช้ในการรายงานผลการทำงานของระบบโดยสรุป

2.1.9 h = histogram("name",n,low,high)

เป็นการกำหนดตาราง ชื่อ "name" ให้พร้อมกับสร้างตารางฮิสโตแกรม ซึ่งมีจำนวนแถวสำหรับการบันทึกข้อมูล n+2 แถว

2.1.10 qtbl = qtable("name")

ใช้สำหรับกำหนดตาราง ชื่อ "name" ขึ้นมาในระบบ เพื่อใช้ในการรายงานผลข้อมูลเกี่ยวกับแถวคอยภายในระบบโดยสรุป

2.1.11 qh = qhistogram("name",n)

เป็นการกำหนดตาราง ชื่อ "name" ให้พร้อมกับสร้างตารางฮิสโตแกรม ซึ่งข้อมูลที่เก็บเกี่ยวกับการกระจายของความยาวของแถวคอย และมีจำนวนแถวสำหรับการบันทึกข้อมูล n+2 แถว

2.2 ประโยคคำสั่งสำหรับการควบคุม

2.2.1 hold(time)

เป็นคำสั่งที่มีผลต่อการปรับค่าเวลาที่จำลองขึ้นในระบบ เพิ่มขึ้นอีกเป็นเวลา time โดย time เป็นข้อมูลชนิด double

2.2.2 attach(f)

บอกความต้องการที่จะเข้าใช้ทรัพยากร f โดยที่ถ้าผู้ให้บริการว่าง กรรมวิธีนั้นเข้าใช้บริการได้ทันที แต่ถ้าผู้ให้บริการ ไม่ว่าง คือมีกรรมวิธีอื่นใช้บริการอยู่ กรรมวิธีที่ต้องการใช้ทรัพยากรนี้ต้องคอยอยู่ในแถวคอยตามลำดับการมาถึง

2.2.3 detach(f)

บอกเลิกการใช้ทรัพยากร f ถ้ามีกรรมวิธีอื่นรอใช้ f อยู่ กรรมวิธีแรกในแถวคอยจะถูกเลือกให้เข้าใช้ทรัพยากร f ต่อไป

2.2.4 use(f,time)

เป็นการเข้าใช้ทรัพยากร f ด้วยเวลา time ซึ่งคำสั่งนี้เป็นผลมาจากการทำงานของ 3 คำสั่งคือ attach(f) , hold(time) และ detach(f) ตามลำดับ

2.2.5 rm_facility(f)

ยกเลิกทรัพยากร f ออกจากระบบที่จำลอง ซึ่งทำให้ทุกกรรมวิธีที่คอยใช้ทรัพยากรนี้ หายไปจากระบบด้วย

2.2.6 st = state(ev)

บอกสถานะของเหตุการณ์ ev ในขณะนั้น ว่าเกิดขึ้นในระบบหรือยัง (Occur or Nonoccur) โดยให้ค่าตัวเลข 0 (= N_OCC) และ 1 (= OCC)

2.2.7 set(ev)

กำหนดให้เหตุการณ์ ev มีสถานะเป็น "Occur" ซึ่งทำให้กรรมวิธีที่คอยการเกิดเหตุการณ์นี้อยู่ ทำงานต่อไปได้ และระบบจะกำหนดสถานะของ ev นี้ให้

เป็น "Nonoccur" ตามเดิม แต่ในกรณีที่ไม่มีกรรมวิธีใดคอยอยู่ เหตุการณ์ ev นั้นยังคงมีสถานะเป็น "Occur"

2.2.8 reset(ev)

กำหนดสถานะของเหตุการณ์ ev ให้เป็น Nonoccur

2.2.9 wait(ev)

เป็นการหยุดรอให้เหตุการณ์ ev มีสถานะเป็น "Occur" ก่อน แล้วทุกกรรมวิธีที่รออยู่ทั้งหมด จึงสามารถดำเนินต่อไปได้

2.2.10 wait_blk(ev)

เป็นการหยุดรอเช่นเดียวกับ wait(ev) แต่ต่างกันตรงที่กรรมวิธีที่จะดำเนินต่อไปนั้น มิได้เพียง 1 กรรมวิธีเท่านั้น

2.2.11 rm_event(ev)

ยกเลิกเหตุการณ์ ev ออกจากระบบ พร้อมกับกรรมวิธีทั้งหมด ที่คอยการเกิดเหตุการณ์นี้

2.2.12 rm_event_array(ev,n)

ยกเลิกแถวลำดับของเหตุการณ์ ev ตั้งแต่ 0 ถึง n-1

2.2.13 mb_st = mbox_state(mb)

บอกสถานะของตู้จดหมาย mb ในขณะนั้น ว่ามีข่าวสารส่งเข้ามาหรือยัง โดยให้ค่าตัวเลขเป็น 0 ถ้ายังไม่มีข่าวสาร และเป็น 1 ถ้ามีข่าวสารใน mb

2.2.14 send(mb,msg)

ส่งข่าวสารซึ่งเป็นตัวเลขจำนวนเต็ม ไปเก็บไว้ในตู้จดหมาย mb

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

2.2.15 receive(mb)

รับข่าวสารที่ส่งมาที่ตู้จดหมาย mb ถ้ายังไม่มีข่าวสารใดๆ กรรมวิธีนั้นต้องรอ จนกว่าจะได้รับข่าวสาร จึงดำเนินการต่อไปได้

2.2.16 rm_mbox(mb)

ยกเลิกตู้จดหมาย mb ออกจากระบบ พร้อมกับกรรมวิธีที่รอข่าวสารในตู้จดหมายนี้ทั้งหมด

2.2.17 rm_mbox_array(mb,n)

ยกเลิกแถวลำดับของตู้จดหมาย mb ตั้งแต่ 0 ถึง n-1

2.2.18 rm_table(tbl)

ยกเลิกตาราง ชื่อ tbl ออกจากระบบ

2.3 <u>ประโยคคำสั่งสำหรับการเก็บรวบรวมข้อมูล</u>

2.3.1 record(tbl,x)

ใช้บันทึกค่า x ไว้ในตารางชื่อ tbl ซึ่งกำหนดด้วยคำสั่ง table โดย x มีชนิดของข้อมูลเป็น double

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

2.3.2 in_qtable(qtbl)

ใช้ปรับข้อมูลเกี่ยวกับสถานะของแถวคอยและความยาวของแถวคอยเพิ่มขึ้น 1

2.3.3 out_qtable(qtbl)

ใช้ปรับข้อมูลเกี่ยวกับสถานะของแถวคอยและความยาวของแถวคอยลดลง 1

2.4 ประโยคคำสั่งสำหรับการสร้างตัวเลขสุ่ม

2.4.1 tm = expdev(m)

เป็นคำสั่งที่ใช้ในการสร้างตัวเลขสุ่มที่มีการกระจายแบบเอกซ์โปเนนเชียล (Exponential distribution) และมีค่าเฉลี่ยเป็น m โดยที่ m มีชนิดของข้อมูลเป็น double และค่าตัวเลขสุ่มที่ได้จากการเรียกใช้คำสั่งนี้เก็บในตัวแปรชนิด double ชื่อ tm

2.4.2 prob()

เป็นการสร้างตัวเลขสุ่มที่มีการกระจายแบบเอกภาพ (Uniform) โดยมีค่าอยู่ในช่วงตั้งแต่ 0 ถึง 1

3. ตัวอย่างโปรแกรมการจำลองระบบแถวคอยแบบ M/M/1

โปรแกรมตัวอย่างนี้เก็บไว้ในแฟ้มข้อมูลชื่อ queue.c จากนั้นทำการแปลโปรแกรมนี้เพื่อให้ได้ queue.o โดยใช้คำสั่ง

```
chulkn>cc -c queue.c
```

ขั้นต่อไปทำการเชื่อม queue.o กับ np.a เพื่อให้ได้ np โดยใช้คำสั่ง

```
chulkn>cc -o np np.a queue.o
```

```
/*        ---------  queue.c  --------    */
/*        simulate an m/m/1 queue
          (an open queue with exponential service times and interarrival
           intervals)
*/
```

```
#include "np.h"

#define     MAXCUST   5000    /* number of customer to be simulated */

double      IATM = 2.0;       /* mean of inter-arrival time */
double      SVTM = 1.0;       /* mean of service time */
double      tm;               /* variable for generated time */
FACILITY    *f;               /* variable for facility */
EVENT       *done;            /* variable for event */
TABLE       *tbl;             /* pointer to table */
QTABLE      *qtbl;            /* pointer for qhistogram */
int         cnt;              /* number of active tasks */

np()
{
    int i;
     gen();                                /* required gen stmt. */
     f     = facility("f1");               /* declare facility */
     done  = event("done");                /* declare event */
     tbl   = table("time tbl");
     qtbl  = qhistogram("Que. time",12); /* create queue histogram */
     cnt   = MAXCUST;
     for (i = 1; i <= MAXCUST; i++)
     {
      tm = expdev(IATM);           /* generate inter-arrival time */
      hold(tm);                    /* hold inter-arrival time */
      cust();                      /* initiate process cust */
     }
     wait(done);                   /* wait until all cust done */
     report();                     /* print report */
}
```

```
cust()                                /* process customer */
{
     gen();                           /* required gen stmt. */
     in_qtable(qtbl);                 /* observe arrival */
     attach(f);                       /* attach facility */
     tm = expdev(SVTM);               /* generate service time */
     hold(tm);                        /* hold service time */
     detach(f);                       /* detach facility */
     record(tbl,&tm);                 /* record time */
     out_qtable(qtbl);                /* observe departure */
     cnt--;                           /* decrement cnt */
     if(cnt == 0)
     {
      /* if last arrival, signal */
        set(done);
     }
}

report()
{
  float rho, n, rt, tp;
     tp  = 1.0 / IATM;
     rho = tp * SVTM;
     n   = rho / (1.0 - rho);
     rt  = SVTM / (1.0 - rho);
     printf("\n\t\t\tM/M/1 Theoritical Result\n");
     printf("Interarrival time  %.3f\n",IATM);
     printf("Mean service time  %.3f\n",SVTM);
     printf("Utilization        %.3f\n",rho);
     printf("Throughput         %.3f\n",tp);
     printf("Response time      %.3f\n",rt);
```

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

```
    printf("Mean no. in system %.3f\n",n);
    printf("Mean queue length  %.3f\n",n - rho);
    printf("Time in queue      %.3f\n",rt - SVTM);
}
```

ผลลัพธ์ที่ได้จากการทำงานของโปรแกรม

```
         M/M/1 Theoritical Result
Interarrival time  2.000
Mean service time  1.000
Utilization        0.500
Throughput         0.500
Response time      2.000
Mean no. in system 1.000
Mean queue length  0.500
Time in queue      1.000

          NP  Simulation Report
Simulated time    9916.183455
No. of process    5001
          Facility Details
Name                       f1
No. of server              1
ID                         0
Utilization                0.5046
Output rate                0.5042
Mean response time         2.1265
Mean service  time         1.0007
Mean population(q. len)    1.0722
Mean no. of busy server    0.5046
---------------------------------------------
```

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

Queue Table Detail

Name	Que. time
mean queue length	1.0000
mean waiting time	2.1265
number of arrivals	5000
number of departures	5000
maximum queue length	12

Queue Table Histogram

length	count	mean-time	% time	cumulative
0	2472	1.987	49.54	49.54
1	3687	0.660	24.53	74.07
2	1832	0.669	12.36	86.43
3	922	0.647	6.01	92.44
4	481	0.687	3.33	95.77
5	262	0.674	1.78	97.55
6	142	0.645	0.92	98.47
7	89	0.837	0.75	99.23
8	53	0.631	0.34	99.56
9	31	0.724	0.23	99.79
10	17	0.880	0.15	99.94
over	12	0.495	0.06	100.00

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

Table Detail

Name	time tbl
number of entries	5000
maximum value	19.0427
minimum value	0.0004
sum of x	6659.6130
sum of x**2	22512.1143
mean of x	1.3319
standard deviation	1.6520

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก ข

รูปแบบของส่วนต้นแฟ้มปฏิบัติการชนิด .EXE

โปรแกรมภาษาเครื่องที่ตัวแปลภาษา Turbo C สร้างขึ้น จะอยู่ในรูปแบบ .COM และ .EXE ซึ่งเป็นแฟ้มปฏิบัติการภายใต้ระบบปฏิบัติการดอส(DOS) โดยที่โปรแกรมที่เป็น .COM สามารถปฏิบัติการได้โดยไม่ต้องปรับข้อมูล สำหรับโปรแกรม .EXE นั้น เนื่องจากแฟ้มปฏิบัติการชนิดนี้สามารถเปลี่ยนแปลงตำแหน่งที่อยู่เริ่มต้นสำหรับปฏิบัติการโปรแกรมได้ จึงจำเป็นต้องมีข้อมูลสำหรับการโหลด (load) โปรแกรมภาษาเครื่องชนิดนี้ โดยข้อมูลนี้จะถูกเก็บไว้ในโครงสร้างที่ซับซ้อนตรงส่วนต้นของแฟ้มปฏิบัติการชนิด .EXE เรียกว่า header record โดยมีรายละเอียดดังนี้

ค่าออฟเซต (แสดงด้วยตัวเลขฐาน 16)	ข้อมูล
00	เป็นข้อมูลบ่งชี้ว่าแฟ้มนี้เป็น .EXE ซึ่งมีค่า $(5A4D)_{16}$
02	เศษจากการหารขนาดแฟ้มด้วย 512
04	จำนวนเพจ (page) บวก 1 โดยที่แต่ละเพจมีขนาด 512 ไบต์
06	จำนวนเอนทรี (entry) ในตารางการย้ายตำแหน่ง (relocation table)
08	จำนวนพารากราฟ (paragraph) ใน header record โดยที่แต่ละพารากราฟมีขนาดเท่ากับ 16 ไบต์
0A	จำนวนพารากราฟน้อยสุดที่ heap และ stack ต้องใช้
0C	จำนวนพารากราฟมากสุดที่ heap และ stack ต้องการ
0E	ค่าขจัด (displacement) สำหรับรีจิสเตอร์สแตกเซกเมนต์ หรือ SS

ค่าออฟเซต (แสดงด้วยตัวเลขฐาน 16)	ข้อมูล
10	ค่าเริ่มต้นของรีจิสเตอร์ตัวชี้สแตก หรือ SP
12	ค่าผลรวม (checksum) ซึ่งโปรแกรม loader ใช้สำหรับตรวจสอบความสมบูรณ์ของแฟ้ม
14	ค่าเริ่มต้นของรีจิสเตอร์ตัวชี้คำสั่ง หรือ IP
16	ค่าขจัดสำหรับรีจิสเตอร์โค้ดเซกเมนต์ หรือ CS
18	ค่าออฟเซตของตารางการย้ายตำแหน่งซึ่งนับเริ่มจากตำแหน่งแรกของ header record
1A	หมายเลขส่วนซ้อนแทน (overlay) ซึ่งถ้ามีค่า 0 หมายถึง เป็นโปรแกรมแบบ standalone หรือเป็นโปรแกรมส่วนเริ่มต้นของระบบซ้อนแทน

ข้อมูลส่วน header record นี้ใช้จำนวนไบต์ขนาดคงที่ 28 ไบต์ และข้อมูลส่วนถัดไปจะเป็นตารางการย้ายตำแหน่ง ซึ่งข้อมูลแต่ละเอนทรีในตารางใช้เนื้อที่ 4 ไบต์โดยที่ 2 ไบต์แรกใช้ปรับค่ารีจิสเตอร์โค้ดเซกเมนต์ และอีก 2 ไบต์ ใช้เป็นค่าออฟเซตของตำแหน่งข้อมูลในโปรแกรมที่จะต้องปรับย้ายตำแหน่งไปตามตำแหน่งที่โปรแกรมนั้นถูกโหลดลงหน่วยความจำ และขนาดของตารางการย้ายตำแหน่งนี้ แปรไปตามจำนวนเอนทรีซึ่งระบุอยู่ใน header record ตำแหน่งออฟเซตที่ $(06)_{16}$

ดังนั้นส่วนสำคัญที่ต้องพิจารณาก่อนเริ่มปฏิบัติการในแบบจำลองคือ จะมีการตรวจสอบจำนวนเอนทรีที่จำเป็นต้องเปลี่ยนแปลงให้ถูกต้องไปตามตำแหน่งที่โปรแกรมถูกโหลดเข้ามา แล้วทำการเปลี่ยนแปลงข้อมูล ณ ตำแหน่งไบต์ออฟเซตที่ระบุในแต่ละเอนทรี ด้วยการรวมค่าในรีจิสเตอร์โค้ดเซกเมนต์ที่ได้ปรับค่าแล้ว (ด้วยค่าออฟเซตตำแหน่งที่ $(16)_{16}$ ของ header record) กับค่าข้อมูล ณ ตำแหน่งนั้น จนครบทุกเอนทรี

จากข้อมูล header record ข้างบนนี้ทำให้สามารถนำแฟ้มปฏิบัติการชนิด .EXE เข้ามาปฏิบัติการโดยตัวแบบจำลองที่พัฒนาได้ด้วยการกำหนดโครงสร้างภายในโปรแกรมดังนี้

```
typedef struct {
    unsigned
        signature,
        modsize,
        npages,
        relocitems,
        headersize,
        minheap,
        maxheap,
        stackdispl,
        initsp,
        checksum,
        entrypoint,
        codeseg,
        reloctable,
        overlaynbr;
} HEADREC;
```

ก่อนเริ่มต้นทำงานของตัวแบบจำลอง ต้องปรับปรุงค่าในรีจิสเตอร์ที่ระบุด้วยข้อมูลใน header record ที่อ่านเข้ามาตามโครงสร้างที่กำหนด ได้แก่

ปรับค่าเริ่มต้นของรีจิสเตอร์สแตกเซกเมนต์ โดยรวมกับค่าขจัดที่อยู่ในตำแหน่งออฟเซต ที่ $(0E)_{16}$ ของ header record หรือค่าตัวแปรในโครงสร้างชื่อ stackdispl

กำหนดค่าเริ่มต้นของรีจิสเตอร์ตัวชี้สแตก ด้วยค่าในตำแหน่งออฟเซตที่ $(10)_{16}$ ของ header record หรือค่าตัวแปรในโครงสร้างชื่อ initsp

กำหนดค่าเริ่มต้นของรีจิสเตอร์ตัวชี้คำสั่ง ด้วยค่าในตำแหน่งออฟเซตที่ $(14)_{16}$ ของ header record หรือค่าตัวแปรในโครงสร้างชื่อ entrypoint

ปรับค่าเริ่มต้นของรีจิสเตอร์โค้ดเซกเมนต์ โดยรวมกับค่าขจัดที่อยู่ในตำแหน่งออฟเซต ที่ $(16)_{16}$ ของ header record หรือค่าตัวแปรในโครงสร้างชื่อ codeseg

ในการดำเนินการทดสอบการทำงานของแบบจำลองที่พัฒนาขึ้นบนเครื่องมินิคอมพิวเตอร์นี้ กำหนดใช้โปรแกรมสำหรับการวัดเปรียบเทียบสมรรถนะทั้ง 5 โปรแกรมซึ่งกล่าวไว้แล้วในบทที่ 4 มาเป็นข้อมูลให้ตัวแปลภาษา Turbo C เวอร์ชัน 1.0 สร้างเป็นโปรแกรมภาษาเครื่องขึ้นมาบนเครื่องคอมพิวเตอร์แบบบุคคล ดังนั้นจึงจำเป็นต้องเข้ารหัส (encode) โปรแกรมภาษาเครื่องเหล่านี้ด้วยคำสั่ง uuencode ซึ่งมีใช้บนเครื่องคอมพิวเตอร์แบบบุคคล แล้วจึงทำการย้ายข้อมูลที่เข้ารหัสไว้แล้วขึ้นไปยังเครื่องมินิคอมพิวเตอร์ VAX-11/750 หรือ VAX-11/785 ที่ใช้ในการดำเนินการทดสอบด้วยโปรแกรม XTALK หรือ PROCOMM หลังจากที่ย้ายข้อมูลเหล่านั้นขึ้นมาแล้ว ก็จะต้องแยกรหัส (decode) โปรแกรมภาษาเครื่องที่ได้เข้ารหัสไว้ให้กลับสู่รูปแบบเดิมโดยใช้คำสั่ง uudecode ซึ่งมีใช้บนเครื่องมินิคอมพิวเตอร์

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก ค

ข้อมูลที่จำเป็นสำหรับการปฏิบัติการคำสั่ง

หน่วยปฏิบัติการใช้เวลาในการปฏิบัติการคำสั่งด้วยจำนวนรอบสัญญาณนาฬิกาที่แตกต่างกันไปตามประเภทคำสั่ง จำนวนตัวถูกกระทำการ และวิธีการอ้างตำแหน่งข้อมูล จึงได้รวบรวมเวลาสำหรับการปฏิบัติการแต่ละคำสั่ง โดยมีรูปแบบการกำหนดให้แต่ละคำสั่งดังนี้

คำสั่ง ประเภทคำสั่ง จำนวนตัวถูกกระทำการ เวลาที่ใช้ปฏิบัติการ

โดยแต่ละส่วนนี้กำหนดความหมายดังนี้

1. คำสั่ง แบ่งแสดงได้ 2 ส่วนคือ
   - 1.1 ตัวพิมพ์ใหญ่ เป็นคำสั่ง
   - 1.2 ตัวพิมพ์เล็ก เป็นตัวถูกกระทำการ ซึ่งมีได้หลายประเภทตามวิธีการอ้างตำแหน่ง ดังที่ได้อธิบายไว้ในบทที่ 4
     - 1.2.1 reg หมายถึงรีจิสเตอร์สำหรับใช้งานในคำสั่ง ซึ่งได้แก่ AX , BX , CX , DX , SI , DI , BP , SP CS , DS , ES และ SS
       - โดยที่ ac คือ รีจิสเตอร์ AX
       - segreg คือ รีจิสเตอร์ CS , DS , ES และ SS เท่านั้น
     - 1.2.2 mem หมายถึงตัวถูกกระทำการมาจากหน่วยความจำหลัก ในบางคำสั่งจำเป็นต้องใช้ข้อมูลถึง 4 ไบต์เพื่อให้มีช่วงการอ้างอิงตำแหน่งในหน่วยความจำหลักได้เพิ่มขึ้น
     - 1.2.3 data หมายถึงค่าข้อมูลของตัวถูกกระทำการ

ศูนย์วิทยทรัพยากร จุฬาลงกรณ์มหาวิทยาลัย

1.2.4 disp หมายถึงค่าขจัดซึ่งต้องนำไปคำนวณตามวิธีการอ้างตำแหน่ง ในบางคำสั่งมีการระบุใช้จำนวนบิตสำหรับขนาดค่าขจัด เช่น 8 บิต หรือ 16 บิต

1.2.5 addr หมายถึงตำแหน่งที่อยู่ของตัวถูกกระทำการ

1.2.6 prefix หมายถึงคำสั่งนำหน้าซึ่งใช้คู่กับคำสั่งปฏิบัติการกับตัวอักษร

1.2.7 type หมายถึงประเภทหรือหมายเลขฟังก์ชันการจัดจังหวะ

บางคำสั่งของคำสั่งคำนวณทางคณิตศาสตร์ได้ถูกกำหนดใช้เวลาในการปฏิบัติการคำสั่งแตกต่างกันตามขนาดตัวถูกกระทำการเช่น reg หรือ mem ซึ่งอาจเป็น 8 บิต หรือ 16 บิต

2. ประเภทคำสั่งกำหนดแทนด้วยหมายเลข

หมายเลข	ประเภทคำสั่ง
0	คำสั่งเคลื่อนย้ายข้อมูล
1	คำสั่งคำนวณทางคณิตศาสตร์
2	คำสั่งปฏิบัติการกับบิต
3	คำสั่งปฏิบัติการกับตัวอักษร
4	คำสั่งย้ายการควบคุม
5	คำสั่งนำหน้า
6	คำสั่งเกี่ยวกับอุปกรณ์นำข้อมูลเข้า/ออก
7	คำสั่งควบคุมตัวประมวลผล

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

3. จำนวนตัวถูกกระทำการ แสดงด้วยตัวเลข ดังนี้

0 หมายถึงไม่มีตัวถูกกระทำการ

1 หมายถึงมีตัวถูกกระทำการ 1 ตัว

2 หมายถึงมีตัวถูกกระทำการ 2 ตัว

4. เวลาที่ใช้ปฏิบัติการ มีหน่วยเป็นรอบสัญญาณนาฬิกา (clock cycles) โดยที่ตัวเลขในวงเล็บ เป็นเวลาที่ใช้ปฏิบัติการเมื่อมีการทำงาน เช่น กระโดดข้ามคำสั่ง หรือเกิดการขัดจังหวะ นอกจากนี้เวลาที่ใช้ปฏิบัติการบางคำสั่งต้องรวมกับค่าเวลาที่ใช้ในการคำนวณค่าตำแหน่งแท้จริงของตัวถูกกระทำการซึ่งค่านี้ขึ้นอยู่กับวิธีการอ้างตำแหน่ง รายละเอียดของค่าเวลาที่ต้องเพิ่มสามารถค้นหาได้จาก (ศิริวรรณ,2534) และ (Rector and Alexy,1980)

รายละเอียดข้อมูลที่ใช้สำหรับการปฏิบัติการคำสั่งของหน่วยปฏิบัติการ แสดงได้ดังนี้

คำสั่ง	ประเภทคำสั่ง	จำนวนตัวถูกกระทำการ	เวลาที่ใช้ปฏิบัติการ
PUSH reg	0	1	10.0
POP reg	0	1	8.0
POP mem	0	1	17.0
ADD reg_1,reg_2	1	2	3.0
ADD mem,reg	1	2	16.0
ADD reg,mem	1	2	9.0
ADD AL,data	1	2	4.0
ADD AX,data	1	2	4.0
OR reg_1,reg_2	2	2	3.0
OR mem,reg	2	2	16.0
OR reg,mem	2	2	9.0
OR ac,data	2	2	4.0
ADC reg_1,reg_2	1	2	3.0
ADC mem,reg	1	2	16.0
ADC reg,mem	1	2	9.0
ADC ac,data	1	2	4.0
SBB reg_1,reg_2	1	2	3.0
SBB mem,reg	1	2	16.0
SBB reg,mem	1	2	9.0

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

คำสั่ง	ประเภทคำสั่ง	จำนวนตัวถูกกระทำการ	เวลาที่ใช้ปฏิบัติการ
SBB ac,data	1	2	4.0
AND reg_1,reg_2	2	2	3.0
AND mem,reg	2	2	16.0
AND reg,mem	2	2	9.0
AND ac,data	2	2	4.0
SEG segreg	5	0	2.0
DAA	1	0	4.0
SUB reg_1,reg_2	1	2	3.0
SUB mem,reg	1	2	16.0
SUB reg,mem	1	2	9.0
SUB ac,data	1	2	4.0
DAS	1	0	4.0
XOR reg_1,reg_2	2	2	3.0
XOR mem,reg	2	2	16.0
XOR reg,mem	2	2	9.0
XOR ac,data	2	2	4.0
AAA	1	0	4.0
CMP reg_1,reg_2	1	2	3.0
CMP mem,reg	1	2	16.0
CMP reg,mem	1	2	9.0
CMP ac,data	1	2	4.0
AAS	1	0	4.0
INC reg	1	1	2.0
DEC reg	1	1	2.0
JO disp	4	1	4.0 (16.0)
JNO disp	4	1	4.0 (16.0)

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

คำสั่ง	ประเภทคำสั่ง	จำนวนตัวถูกกระทำการ	เวลาที่ใช้ปฏิบัติการ
JB/JNAE/JC disp	4	1	4.0 (8.0)
JNB/JAE/JNC disp	4	1	4.0 (16.0)
JE/JZ disp	4	1	4.0 (16.0)
JNE/JNZ disp	4	1	4.0 (16.0)
JBE/JNA disp	4	1	4.0 (16.0)
JNBE/JA disp	4	1	4.0 (16.0)
JS disp	4	1	4.0 (16.0)
JNS disp	4	1	4.0 (16.0)
JP/JPE disp	4	1	4.0 (16.0)
JNP/JPO disp	4	1	4.0 (16.0)
JL/JNGE disp	4	1	4.0 (16.0)
JNL/JGE disp	4	1	4.0 (16.0)
JLE/JNG disp	4	1	4.0 (16.0)
JNLE/JG disp	4	1	4.0 (16.0)

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

คำสั่ง	ประเภทคำสั่ง	จำนวนตัวถูกกระทำการ	เวลาที่ใช้ปฏิบัติการ
ADD reg,data	1	2	4.0
ADD mem,data	1	2	17.0
OR reg,data	2	2	4.0
OR mem,data	2	2	17.0
ADC reg,data	1	2	4.0
ADC mem,data	1	2	17.0
SBB reg,data	1	2	4.0
SBB mem,data	1	2	17.0
AND reg,data	2	2	4.0
AND mem,data	2	2	17.0
SUB reg,data	1	2	4.0
SUB mem,data	1	2	17.0
XOR reg,data	2	2	4.0
XOR mem,data	2	2	17.0
CMP reg,data	1	2	4.0
CMP mem,data	1	2	10.0
TEST reg_1,reg_2	2	2	3.0
TEST mem,reg	2	2	9.0
XCHG reg_1,reg_2	0	2	4.0
XCHG mem,reg	0	2	17.0
MOV reg_1,reg_2	0	2	2.0
MOV mem,reg	0	2	9.0
MOV reg,mem	0	2	8.0
MOV reg,segreg	0	2	2.0
MOV mem,segreg	0	2	9.0
LEA reg,addr	0	2	2.0
MOV segereg,reg	0	2	2.0
MOV segreg,mem	0	2	8.0

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

คำสั่ง	ประเภทคำสั่ง	จำนวนตัวถูกกระทำการ	เวลาที่ใช้ปฏิบัติการ
NOP	7	0	3.0
XCHG ac,reg	0	2	3.0
CBW	1	0	2.0
CWD	1	0	5.0
CALL addr	4	1	28.0
WAIT	7	0	0.0
PUSHF	0	0	10.0
POPF	0	0	8.0
SAHF	0	0	4.0
LAHF	0	0	4.0
MOV ac,addr	0	2	10.0
MOV addr,ac	0	2	10.0
MOVS	3	0	18.0
MOVS	3	0	17.0
CMPS	3	0	22.0
TEST ac,data	2	2	4.0
STOS	3	0	11.0
STOS	3	0	10.0
LODS	3	0	12.0
LODS	3	0	13.0
SCAS	3	0	15.0
MOV reg,data	0	2	4.0
RET disp (ขนาด 16 บิต)	4	1	20.0
RET (ภายในเซกเมนต์)	4	0	16.0
LES reg,addr	0	2	16.0
LDS reg,addr	0	2	16.0
MOV mem,data	0	2	10.0
RET disp (ขนาด 16 บิต)	4	1	23.0

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

คำสั่ง	ประเภทคำสั่ง	จำนวนตัวถูกกระทำการ	เวลาที่ใช้ปฏิบัติการ
RET (ระหว่างเซกเมนต์)	4	0	24.0
INT 3	4	1	52.0
INT type	4	1	51.0
INTO	4	0	4.0 (53.0)
IRET	4	0	32.0
ROR/ROL reg,1	2	2	2.0
ROR/ROL mem,1	2	2	15.0
ROR/ROL reg,CL	2	2	8.0
ROR/ROL mem,CL	2	2	20.0
RCR/RCL reg,1	2	2	2.0
RCR/RCL mem,1	2	2	15.0
RCR/RCL reg,CL	2	2	8.0
RCR/RCL mem,CL	2	2	20.0
SHR/SHL reg,1	2	2	2.0
SHR/SHL mem,1	2	2	15.0
SHR/SHL reg,CL	2	2	8.0
SHR/SHL mem,CL	2	2	20.0
SAR/SAL reg,1	2	2	2.0
SAR/SAL mem,1	2	2	15.0
SAR/SAL reg,CL	2	2	8.0
SAR/SAL mem,CL	2	2	20.0
AAM	1	0	83.0
AAD	1	0	60.0
XLAT	0	0	11.0
ESC reg	7	1	2.0
ESC mem	7	1	8.0

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

คำสั่ง	ประเภทคำสั่ง	จำนวนตัวถูกกระทำการ	เวลาที่ใช้ปฏิบัติการ
LOOPNE/LOOPNZ disp	4	1	5.0
			(19.0)
LOOPE/LOOPZ disp	4	1	6.0
			(18.0)
LOOP disp	4	1	5.0
			(17.0)
JCXZ disp	4	1	6.0
			(18.0)
CALL disp (ขนาด 16 บิต)	4	1	19.0
JMP disp	4	1	15.0
JMP addr	4	1	15.0
LOCK	7	0	2.0
REP	5	0	2.0
HLT	7	0	2.0
CMC	7	0	2.0
TEST reg,data	2	2	5.0
TEST mem,data	2	2	11.0
NOT/NEG reg	2	1	3.0
NOT/NEG mem	2	1	16.0
MUL reg (ขนาด 8 บิต)	1	1	70.0
MUL mem (ขนาด 8 บิต)	1	1	76.0
IMUL reg (ขนาด 8 บิต)	1	1	80.0
IMUL mem (ขนาด 8 บิต)	1	1	86.0
DIV reg (ขนาด 8 บิต)	1	1	80.0
DIV mem (ขนาด 8 บิต)	1	1	86.0
IDIV reg (ขนาด 8 บิต)	1	1	101.0
IDIV mem (ขนาด 8 บิต)	1	1	107.0

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

คำสั่ง	ประเภทคำสั่ง	จำนวนตัวถูกกระทำการ	เวลาที่ใช้ปฏิบัติการ
MUL reg (ขนาด 16 บิต)	1	1	118.0
MUL mem (ขนาด 16 บิต)	1	1	124.0
IMUL reg (ขนาด 16 บิต)	1	1	128.0
IMUL mem (ขนาด 16 บิต)	1	1	134.0
DIV reg (ขนาด 16 บิต)	1	1	144.0
DIV mem (ขนาด 16 บิต)	1	1	150.0
IDIV reg (ขนาด 16 บิต)	1	1	165.0
IDIV mem (ขนาด 16 บิต)	1	1	171.0
CLC	7	0	2.0
STC	7	0	2.0
CLI	7	0	2.0
STI	7	0	2.0
CLD	7	0	2.0
STD	7	0	2.0
INC/DEC reg	1	1	3.0
INC/DEC mem	1	1	15.0
CALL reg	4	1	16.0
CALL mem	4	1	21.0
CALL mem (ขนาด 32 บิต)	4	1	37.0
JMP reg	4	1	11.0
JMP mem	4	1	16.0
JMP mem (ขนาด 32 บิต)	4	1	24.0
PUSH reg	0	1	11.0
PUSH mem	0	1	16.0

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย

ประวัติผู้เขียน

นางสาวดารารัตน์ แซ่ลี้ เกิดเมื่อวันที่ 13 พฤศจิกายน 2507 จบการศึกษาปริญญาตรีวิทยาศาสตรบัณฑิต สาขาคณิตศาสตร์ จากมหาวิทยาลัยสงขลานครินทร์ เมื่อปีการศึกษา 2528 เข้าศึกษาระดับปริญญาโทวิทยาศาสตรมหาบัณฑิต สาขาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2530 ปัจจุบันรับราชการตำแหน่งอาจารย์ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์

ศูนย์วิทยทรัพยากร
จุฬาลงกรณ์มหาวิทยาลัย