ภาคผนวก D
ซอฟท์แวร์ (Software)

องค์ประกอบหลัก 3 ส่วนในการประมวลผลข้อมูลด้วยเครื่องคอมพิวเตอร์ คือ

1. **ฮาร์ดแวร์** (Hardware) หมายถึงตัวเครื่องคอมพิวเตอร์และอุปกรณ์ทุกอย่างที่ประกอบกันเป็นระบบคอมพิวเตอร์

2. **ซอฟท์แวร์** (Software) หมายถึงคำสั่งและระบบควบคุมการทำงานทั้งหมดที่ไม่ใช่ฮาร์ดแวร์ ซึ่งคือโปรแกรมชนิดต่าง ๆ นั่นเอง

3. **พีเพิลแวร์** (Peopleware) หมายถึงบุคลากรในศูนย์คอมพิวเตอร์ทุกระดับ

เราได้รู้จักกับฮาร์ดแวร์มาแล้ว ในตอนต่อไปนี้เราจะศึกษาถึงเรื่องซอฟท์แวร์บ้าง ในสมัยเริ่มแรกที่ใช้คอมพิวเตอร์นั้น โปรแกรมที่สั่งให้เครื่องทำงานตามต้องการนั้นจะต้องเขียนเป็นภาษาเครื่อง (Machine language) ซึ่งอยู่ในเลขฐานสอง ซึ่งยุ่งยากทั้งการเขียนและการตรวจแก้เมื่อมีข้อผิดพลาด จึงได้มีการพัฒนาทางภาษามาเรื่อย ๆ เป็นภาษาในระดับต่ำ (Low-order language) และเป็นภาษาในระดับสูง (High-order language) ซึ่งเป็นภาษาที่เราคุ้นเคยและเรียนรู้กันแพร่หลายในปัจจุบัน เช่น ภาษา COBOL ภาษา FORTRAN ภาษา BASIC เป็นต้น ภาษาแต่ละภาษาก็เหมาะสมกับงานแต่ละด้าน ไม่มีภาษาใดเพียงภาษาเดียวที่จะเหมาะกับงานทุกชนิดหรือความต้องการทุกรูปแบบ

ถึงแม้ว่าเราจะใช้ภาษาใน Low-order หรือใน High-order ก็ตาม แต่จริง ๆ แล้วเครื่องคอมพิวเตอร์สามารถทำงานตามคำสั่งที่อยู่ในภาษาเครื่องได้เท่านั้น แต่การที่เราสั่งคอมพิวเตอร์เป็นภาษาอื่นนอกจากภาษาเครื่อง แล้วคอมพิวเตอร์สามารถทำตามได้ก็เนื่องจากว่ามีโปรแกรมชนิดหนึ่งซึ่งทำหน้าที่เป็นล่าม รับโปรแกรมที่ไม่ใช่ภาษาเครื่องเป็นข้อมูลเข้า แล้วแปลออกมาเป็นภาษาเครื่อง โดยมีชื่อเรียกเฉพาะดังนี้

1. โปรแกรมที่ไม่ใช่ภาษาเครื่อง เราเรียกว่า **Source program**

2. โปรแกรมที่อยู่ในภาษาเครื่องเราเรียกว่า **Object program**

โปรแกรมล่าม อาจแบ่งออกเป็น 3 ชนิด คือ

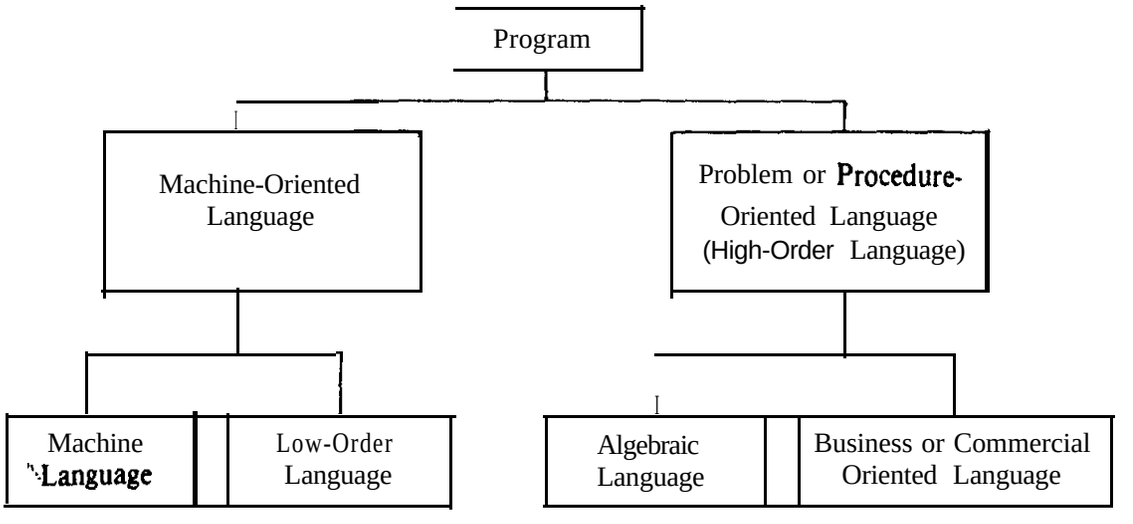
1. Assembler ทำหน้าที่แปลคำสั่งที่เขียนด้วยภาษาใน Low-order ไปเป็นภาษาเครื่อง

2. Compiler ทำหน้าที่แปลคำสั่งที่เขียนด้วย ภาษาใน High-order ไปเป็นภาษาเครื่อง

3. Translator ทำหน้าที่แปลคำสั่งทีละคำสั่ง แล้วเครื่องจะปฏิบัติตามทันทีก่อนที่จะแปลคำสั่งถัดไป การแปลไม่ได้ Object program

โปรแกรมล่ามอาจเรียกในภาษาอังกฤษต่างกันตามหน้าที่ของมัน โปรแกรม Assembler จะแปลคำสั่งที่เขียนด้วยภาษาใน Low-order ไปเป็นคำสั่งในภาษาเครื่องโดยแปล 1 คำสั่งใน Low-order เป็น 1 คำสั่งในภาษาเครื่อง ส่วนโปรแกรม Compiler ทำหน้าที่แปลคำสั่งที่เขียนด้วยภาษาใน High-order ไปเป็นคำสั่งในภาษาเครื่อง โดยแปล 1 คำสั่งใน High-order ไปเป็นหลาย ๆ คำสั่งในภาษาเครื่อง โปรแกรมล่ามทั้ง 2 ชนิดดังกล่าวข้างต้นจะแปล Source program ออกมาเป็น Object program ส่วนล่ามอีกชนิดหนึ่งที่ใช้กับภาษาในระดับสูงบางภาษา คือโปรแกรม Translator ซึ่งจะแปลคำสั่งทีละ 1 คำสั่ง ให้เป็น Object code แล้วเครื่องจะปฏิบัติตามทันทีก่อนที่จะแปลคำสั่งถัดไป ดังนั้น การแปลด้วย Transtator จะไม่ให้ Object program

1 ชนิดของภาษาที่ใช้ในการเขียนโปรแกรม



เราอาจแบ่งภาษาที่ใช้ในการเขียนโปรแกรม ออกเป็น 2 ชนิด คือ

1. **ภาษาที่ขึ้นอยู่กับเครื่อง** (Machine-Oriented Language) เป็นภาษาที่ก่อนจะเขียนได้ผู้เขียนต้องศึกษารายละเอียดการทำงานของเครื่องก่อน ดังนั้น คำสั่งหรือโปรแกรมที่เขียนขึ้นสำหรับสั่งคอมพิวเตอร์เครื่องหนึ่งของบริษัทหนึ่งจะนำไปใช้กับเครื่องคอมพิวเตอร์ของบริษัทอื่นหรือเครื่องที่ต่างรุ่นกันของบริษัทเดียวกันไม่ได้ (บริษัทในที่นี้หมายถึงบริษัทผู้ผลิต)

2. ภาษาที่ขึ้นอยู่กับปัญหาหรือวิธีการ (Problem-Oriented Language หรือ Procedure-Oriented Language) ภาษาชนิดนี้ส่วนมากสามารถสั่งเครื่องคอมพิวเตอร์ได้ทั่ว ๆ ไป แต่อาจมีรายละเอียดแตกต่างกันบ้างเพียงเล็กน้อยสำหรับเครื่องที่ผู้ผลิตต่างกันเป็นภาษาในระดับสูง ภาษาที่จัดเป็น Problem-Oriented นั้น จะเหมาะกับปัญหาเฉพาะเจาะจงลงไป ส่วน Procedure-Oriented Language นั้น เป็นภาษาที่เน้นวิธีหรือการกระทำจริง ๆ

ภาษาที่เป็น Machine-Oriented นั้นแยกเป็นภาษาเครื่อง (Machine Language) และภาษาในระดับต่ำ (Low-Order Language or Symbolic Language)

ภาษาที่เป็น Problem หรือ Procedure Oriented Language อาจแยกเป็น 2 ลักษณะ คือ กลุ่มภาษาที่ใช้ในการคำนวณทางวิทยาศาสตร์-วิศวกรรมศาสตร์ เราเรียกว่า Algebraic Language หรืออาจเรียกว่า Nonbusiness-Oriented Language เช่น ภาษา FORTRAN, ALGOL, APL เป็นต้น อีกลักษณะหนึ่งคือภาษาที่ใช้ในทางธุรกิจ เราเรียกว่า Business หรือ Commercial-Oriented Language เช่น ภาษา COBOL, RPG เป็นต้น ส่วนภาษาที่เป็นทั้ง Business และ Nonbusiness-Oriented Language เช่น BASIC, PL/I และ Pascal เป็นต้น

นอกจากการแบ่งชนิดของภาษาตามรูปข้างต้นแล้ว เราอาจแบ่งภาษาเป็นภาษาที่เป็น General purpose ซึ่งหมายถึงภาษาที่ใช้กับการแก้ปัญหาได้หลาย ๆ ชนิด และภาษาที่เป็น Special purpose ซึ่งเป็นภาษาที่ใช้ได้กับปัญหาเฉพาะอย่างหรือนำไปประยุกต์ใช้กับงานบางอย่างเท่านั้น นอกจากนั้นยังอาจแบ่งเป็นภาษาที่ใช้กับระบบแบช (Batch) และภาษาที่ใช้กับระบบเรียลไทม (Real-time) ระบบแบชคือระบบการประมวลผลข้อมูลซึ่งจะทำการรวบรวมข้อมูลเข้าไว้ระยะเวลาหนึ่งก่อนแล้วจึงทำการประมวลผลพร้อม ๆ กัน ส่วนระบบเรียลไทมนั้นจะทำการประมวลผลข้อมูลทันทีทันใดที่รับข้อมูลเข้าและส่งข้อมูลออกมาให้ทันทีที่ประมวลผลข้อมูลเสร็จแล้ว

2 รายละเอียดของภาษาต่าง ๆ อย่างสังเขป

2.1 ภาษาเครื่อง (Machine language) เป็นภาษาที่ใช้ตัวเลขทั้งหมดคือเลขในระบบเลขฐานสอง มาประกอบกันเป็นคำสั่ง ภาษานี้ต้องใช้เวลาศึกษาวิธีการเขียนมากเพราะต้องศึกษาเรื่องเครื่องพร้อม ๆ กับศึกษาปัญหาที่เราต้องการแก้ด้วย โปรแกรมที่เขียนในภาษานี้เมื่อเกิดข้อผิดพลาดจะแก้ไขได้ยาก และในการเขียนอาจเกิดข้อผิดพลาดได้ง่ายถ้าไม่ชำนาญ

รูปแบบของคำสั่งในภาษาเครื่องคือ

Operation code หรือ op-code	Operand 1 หรือ P-address	Operand 2 หรือ Q-address

โดยที่ Operation code เป็นตัวเลขแสดงรหัสในการปฏิบัติงาน เช่น การอ่าน การบันทึกข้อมูล การคำนวณ เป็นต้น

Operand 1 คือตัวถูกกระทำตัวที่ 1 (หรือ P-address) หมายถึงเลขที่อยู่ของที่ภายในหน่วยความจำซึ่งเก็บข้อมูลหรือคำสั่งที่จะถูกนำไปใช้กับรหัสในการปฏิบัติงานที่ระบุไว้ หรืออาจหมายถึงที่ ๆ เตรียมไว้เก็บข้อมูลก็ได้

Operand 2 คือตัวถูกกระทำตัวที่ 2 (หรือ Q-address) หมายถึงเลขที่อยู่ของที่ภายในหน่วยความจำซึ่งเก็บข้อมูลที่จะถูกนำไปใช้กับรหัสในการปฏิบัติงานที่ระบุไว้

ภาษาเครื่องมีคำสั่งที่เป็นตัวเลขทั้งหมด ซึ่งถ้าเครื่องมีรหัสในการปฏิบัติงาน (Operation code) เป็นจำนวนมากก็เป็นการยากที่จะจดจำได้หมด นอกจากนั้นผู้เขียนยังต้องระมัดระวังในเรื่องของเลขที่อยู่หรือ address ของที่ต่าง ๆ ที่เราจะใช้ในโปรแกรมด้วย

6.2.2 ภาษาในระดับต่ำ (Low-order language หรือ Symbolic language) เป็นภาษาที่สูงขึ้นกว่าภาษาเครื่อง ในคำสั่งแทนที่จะใช้เป็นตัวเลข เราจะใช้ตัวอักษรแทนคำสั่งในการปฏิบัติงาน ซึ่งเรียกว่า นีโมนิคโค้ต (Mnemonic code) และใช้Symbolic name แทน address ของที่ภายในหน่วยความจำหลัก รูปแบบของคำสั่งยังคงใกล้เคียงกับภาษาเครื่อง เพียงแต่สิ่งที่ใช้ในคำสั่งเริ่มมีความหมายใกล้เคียงกับภาษาที่ใช้ในชีวิตประจำวัน เช่น คำสั่งให้อ่าน แทนที่จะใช้รหัสในการปฏิบัติงานเป็นตัวเลข เราจะใช้คำสั่ง เช่น READ หรือ R แทน คำสั่งในการบวกเราจะใช้ ADD หรือ A เป็นต้น

ตัวอย่างเปรียบเทียบรหัสในการปฏิบัติงานในภาษาเครื่องและในภาษาในระดับต่ำ

การปฏิบัติ (Operation)	รหัสในการปฏิบัติงาน(Operation code)	
	ภาษาเครื่อง	ภาษา Assembly
บรรจุ (Load)	10	L
เก็บ (Store)	11	ST
บวก (Add)	20	A
ลบ (Subtract)	21	S

จากตารางเป็นการเปรียบเทียบรหัสในการปฏิบัติงานในภาษาเครื่องและภาษาในระดับต่ำภาษาหนึ่งคือภาษา Assembly ซึ่งใช้กับเครื่องของบริษัท IBM รหัสในภาษาเครื่องนั้นในตารางเป็นเลขฐานสิบแต่เวลาเก็บในตัวเครื่องคอมพิวเตอร์จะเป็นเลขฐานสอง ส่วนในภาษา Assembly นั้นเราใช้ตัวอักษรแทนตัวเลข ซึ่งมักใช้ตัวย่อของการปฏิบัติที่ต้องการ ซึ่งทำให้ง่ายแก่การจดจำได้ดีกว่ารหัสตัวเลข

ตารางแสดงการเปรียบเทียบคำสั่งที่เขียนด้วยภาษาเครื่องและภาษาในระดับต่ำกับคำสั่งในภาษาในระดับสูง

ภาษาในระดับสูง FORTRAN : NETPAY = INCOME – TAX

COBOL : SUBTRACT TAX FROM INCOME GIVING NETPAY.

ภาษาเครื่อง	ภาษา Assembly	คำอธิบาย
10 2 632	L 2, INCOME	บรรจุ Income เข้าไปใน Accumulator (ใช้ที่เลขที่ 2)
21 2 438	S 2, TAX	ลบTAX ออกจาก Accumulator
11 2 9A7	ST 2, NETPAY	เก็บค่าจาก Accumulator ใน NETPAY

จากคำสั่งข้างต้น NETPAY, INCOME และ TAX เป็น Symbolic name หรือ data name คือชื่อของที่ภายในหน่วยความจำหลักซึ่งระบบควบคุมการทำงานของเครื่องคอมพิวเตอร์จะจัดหาที่ให้ โดยเราไม่จำเป็นต้องทราบว่าที่เหล่านั้นอยู่ส่วนใดของหน่วยความจำ เราใช้

Symbolic name ในภาษาในระดับต่ำและระดับสูง ส่วนในภาษาเครื่องเราต้องทราบเลขที่อยู่ของที่ ๆ เก็บข้อมูลที่เราจะนำมาประมวลผลตามคำสั่ง

จากตัวอย่าง

เลขที่อยู่ ของที่ ๆ เก็บค่า INCOME คือ 632 ในฐาน 16

เลขที่อยู่ ของที่ ๆ เก็บค่า TAX คือ 438 ในฐาน 16 และ

เลขที่อยู่ ของที่ ๆ เก็บค่า NETPAY คือ 9A7 ในฐาน 16

ในภาษาเครื่อง เลข 10, 21 และ 11 คือรหัสในการปฏิบัติการ ส่วนภาษาในระดับต่ำใช้ L, S และ ST แทนการบรรจุ การลบและการบันทึกข้อมูลตามลำดับ เลข 2 แทนรียิสเตอร์ (Register) หมายเลข 2 ซึ่งหมายถึง Accumulator ในหน่วยคำนวณและตรรกะ (ALU) ใน CPU นั่นเอง (โดยทั่ว ๆ ไปรียิสเตอร์หมายถึงส่วนของหน่วยความจำซึ่งจะใช้ทำหน้าที่พิเศษตามความต้องการของโปรแกรมเมอร์)

2.3 ภาษาในระดับสูง (High-order language) เป็นภาษาที่ได้รับการพัฒนาแล้วเพื่อให้ใกล้เคียงกับภาษาที่ใช้ในชีวิตประจำวัน ลดความยุ่งยากในการเขียนคำสั่งลงจากคำสั่งที่สั่งด้วยภาษาใน 2 ระดับข้างต้น ทำให้เป็นภาษาที่ใช้ได้ง่าย สะดวกในการตรวจสอบและแก้ไขข้อผิดพลาด ภาษาส่วนใหญ่ที่เราได้ยินในปัจจุบัน เช่น ภาษาฟอร์แทรน ภาษาโคบอล ภาษาปาสกาล ภาษาอาร์พีจี เป็นต้น ล้วนเป็นภาษาในระดับสูงทั้งสิ้น จะได้กล่าวถึงภาษาแต่ละภาษาพอสังเขป และจะมีตัวอย่างโปรแกรมที่เขียนด้วยภาษาในระดับสูงด้วย

2.3.1 ภาษาเบสิก (BASIC)

คำว่า BASIC ย่อมาจาก Beginners All-purpose Symbolic Instruction Code

ภาษาเบสิกเป็นภาษา general purpose, precedure-oriented และนิยมใช้กับระบบเรียลไทม ใช้ได้ทั้งงานทางธุรกิจและไม่ใช่ธุรกิจ ภาษานี้มักจะใช้กับมินิและไมโครคอมพิวเตอร์ เป็นภาษาที่ได้รับความนิยมมากขึ้นเรื่อย ๆ ภาษาเบสิกที่เขียนกับเครื่องคอมพิวเตอร์ต่างกันจะมีหลักภาษาต่างกันมาก แต่ในการเขียนคำสั่งเราไม่ต้องศึกษาตัวเครื่อง เพียงแต่ศึกษาหลักภาษาของเครื่องนั้น ๆ เท่านั้น ดังนั้น โปรแกรมที่เขียนกับคอมพิวเตอร์เครื่องหนึ่งไม่อาจนำมาใช้กับอีกเครื่องหนึ่งได้นอกเสียจากว่าได้ดัดแปลงแก้ไขแล้ว แต่อย่างไรก็ดีภาษานี้ก็เป็นภาษาหนึ่งที่ได้รับความนิยมมากโดยเฉพาะเมื่อใช้กับคอมพิวเตอร์ส่วนตัว (Personal Computer) นอกจากนั้นสถาบันมาตรฐานแห่งชาติของสหรัฐอเมริกา (American National Standards Institute หรือเรียกย่อ ๆ ว่า ANSI) ก็ได้พัฒนาภาษาเบสิกมาตรฐานขึ้นมา

ตัวอย่างโปรแกรมซึ่งเขียนด้วยภาษาเบสิก

```
05   LET T = 0
10   FOR I = 1 TO 15
20   READ X
30   LET T = T+X
40   NEXT I
50   PRINT "THE TOTAL IS"; T
60   DATA 16, 18, 4, 8, 9
70   DATA 4, 1, 3, 7, 41
80   DATA 6, 21, 36, 44, 15
99   END
RUN
THE TOTAL IS 233
TIME .4 sec
READY
```

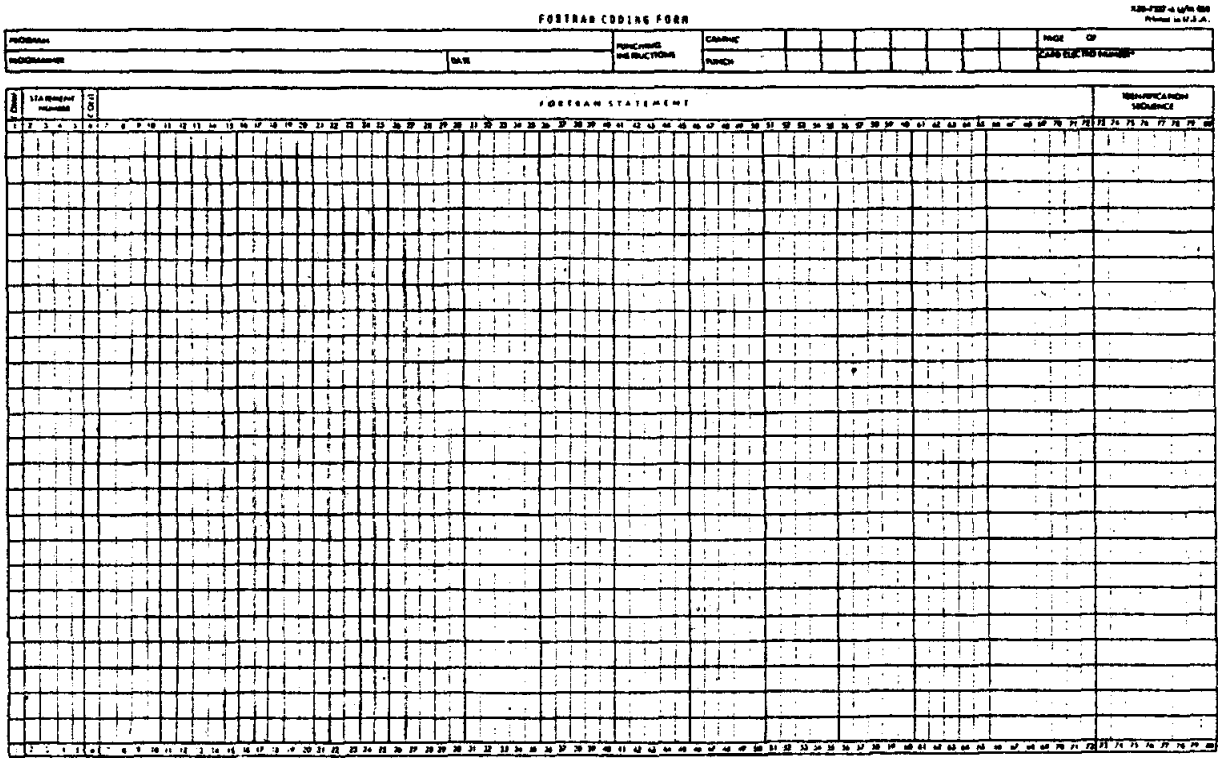
จากตัวอย่างเป็นโปรแกรมในภาษาเบสิก ซึ่งทำการหาผลบวกของเลข 15 จำนวน คือ 16, 18,...,44,15 (ดูในคำสั่ง 60, 70 และ 80) แต่ละคำสั่ง (statement) จะมีหมายเลขประจำคำสั่ง และมีคีย์เวิร์ด (key word) ซึ่งมักตามด้วยการกระทำ จากตัวอย่างตัวเลขในคอลัมน์หน้าสุดคือหมายเลขประจำคำสั่ง (statement number) คีย์เวิร์ดเช่น LET, FOR, READ เป็นต้น การกระทำเช่น T=O, T=T+X เป็นต้น คำสั่ง RUN เป็นคำสั่งให้คอมพิวเตอร์ปฏิบัติตามคำสั่งในโปรแกรมข้างต้น บรรทัดถัดไปคือ THE TOTAL IS 233 เป็นข้อมูลออก (output data) ซึ่งคือผลบวกของเลข 15 จำนวนที่ปรากฏในคำสั่ง DATA ทั้ง 3 บรรทัด

2.3.2 ภาษาฟอร์แทรน (FORTRAN)

คำว่า FORTRAN ย่อมาจาก Formular Translator

ภาษาฟอร์แทรนเป็นภาษา general purpose, procedure-oriented แต่เดิมภาษานี้ถูกพัฒนาเพื่อทำงานในระบบแบช แต่ปัจจุบันสามารถทำงานในระบบเรียลไทมได้ ถึงแม้ว่าภาษาฟอร์แทรนนี้จะถูกสร้างขึ้นเพื่อแก้ปัญหาทางวิทยาศาสตร์ แต่มันก็สามารถใช้กับงานทางด้านธุรกิจได้เช่นกัน

ภาษาฟอร์แทรนเริ่มพัฒนาขึ้นมาโดยบริษัท IBM ประเทศสหรัฐอเมริกาในปี ค.ศ. 1957 จากนั้นเป็นต้นมาภาษาฟอร์แทรนก็มีหลายรุ่น (version) เช่น BASIC FORTRAN, FORTRAN II, FORTRAN IV, FORTRAN V, FORTRAN 77 เป็นต้น เนื่องจากภาษานี้เป็นภาษาที่ถูกสร้างขึ้นเพื่อใช้กับงานทางวิทยาศาสตร์เป็นภาษาแรก มันยังคงครองความนิยมอยู่มาก และสามารถใช้ได้กับระบบคอมพิวเตอร์จำนวนมากด้วย ANSI ได้พัฒนาภาษาฟอร์แทรนมาตรฐานขึ้นเมื่อ ค.ศ.1966 เรียกว่า ANS FORTRAN ทำให้สามารถใช้โปรแกรมที่เขียนด้วยภาษา ANS FORTRAN กับเครื่องหลาย ๆ ระบบได้โดยไม่ต้องแก้ไขเปลี่ยนแปลงเลย การเขียนคำสั่งในภาษาฟอร์แทรนต่างจากภาษาเบสิกตรงที่คำสั่งในภาษาฟอร์แทรนต้องเขียนในตำแหน่งที่เฉพาะเจาะจง กระดาษที่มีตำแหน่งที่บอกไว้เพื่อความสะดวกในการเขียนคำสั่งในแต่ละภาษาเราเรียกว่า Coding form



Coding form สำหรับภาษาฟอร์แทรน

ในภาพเป็น FORTRAN Coding form ของบริษัท IBM ในภาษาฟอร์แทรน เลขที่ประจำคำสั่งหรือเลขหมายประจำคำสั่งนั้นไม่บังคับ เราจะใส่มันให้แก่คำสั่งก็ต่อเมื่อเราต้องการอ้างถึงคำสั่งนั้น ถ้ามีเลขหมายประจำคำสั่งเราต้องใส่ไว้ในคอลัมน์ 1-5 คำสั่งจะเริ่มเขียนจากคอลัมน์ 7 ถึง 72 คอลัมน์ 73-80 จะไม่ถูกแปลโดยฟอร์แทรนคอมไพเลอร์ เรามักใช้สำหรับใส่ชื่อโปรแกรมและเลขลำดับบรรทัดในโปรแกรม

ตัวอย่างโปรแกรมซึ่งเขียนด้วยภาษาฟอร์แทรน

```
คอลัมน์ 1 2 3 4 5 | 6 | 7
                  |   | TOTAL = 0
                  |   | DO 60   I = 1, 15
                  |   | READ (5, 10) X
               10 |   | FORMAT (F8.2)
                  |   | TOTAL = TOTAL+X
               60 |   | CONTINUE
                  |   | WRITE (6,20) TOTAL
               20 |   | FORMAT (1X, 12HTHE TOTAL IS, F15.2)
                  |   | STOP
                  |   | END
```

จากตัวอย่างเป็นโปรแกรมซึ่งเขียนด้วยภาษาฟอร์แทรน 4 ซึ่งเป็นversionหนึ่งของภาษาฟอร์แทรน เป็นคำสั่งให้หาผลบวกของเลข 15 จำนวนซึ่งจะถูกอ่านจากตัวกลางทีละ 1 จำนวน ผลบวกของเลขทั้ง 15 จำนวนจะถูกเก็บใน TOTAL ให้นักศึกษาสังเกตว่าคำสั่งคล้ายคลึงกับคำสั่งในภาษาเบสิกมาก

ปัจจุบันมีภาษาฟอร์แทรน 77 ซึ่งมีประสิทธิภาพสูงกว่าฟอร์แทรน 4 เพราะสามารถทำงานได้เหมือนภาษาฟอร์แทรน 4 และเพิ่มความสามารถในการจัดการกับข้อมูลที่เป็นตัวอักขระได้ด้วย สามารถใช้ในการเขียนโปรแกรมแบบโครงสร้าง (Structured program) ได้ (ภาษาอื่น ๆ ที่ใช้เขียนโปรแกรมแบบโครงสร้างได้ เช่น ภาษาพีแอล 1 ภาษาปาสกาล ภาษาโคบอล เป็นต้น)

2.3.3 ภาษาโคบอล (COBOL)

คำว่า COBOL ย่อมาจาก Common Business Oriented Language

ภาษาโคบอลเป็น business-oriented language ถูกพัฒนาครั้งแรกในปี ค.ศ.1959 ในสหรัฐ หลังจากออกมาใช้แล้วก็มีการพัฒนาอีกหลายครั้ง ในปี ค.ศ.1968 สถาบันมาตรฐานแห่งชาติของสหรัฐอเมริการ่วมกับบริษัทผู้ผลิตเครื่องคอมพิวเตอร์อีกหลายบริษัทได้พัฒนา ANS COBOL (American National Standard COBOL) ขึ้น และใน ค.ศ.1974 ก็พัฒนา ANS COBOL รุ่นใหม่ขึ้นมาอีก ซึ่งถ้าการเขียนโปรแกรมเป็นไปตามกฎเกณฑ์ของมัน เราสามารถนำโปรแกรมไปวิ่ง (run) กับคอมพิวเตอร์ที่มี ANS COBOL คอมไพเลอร์ได้ทุกระบบ

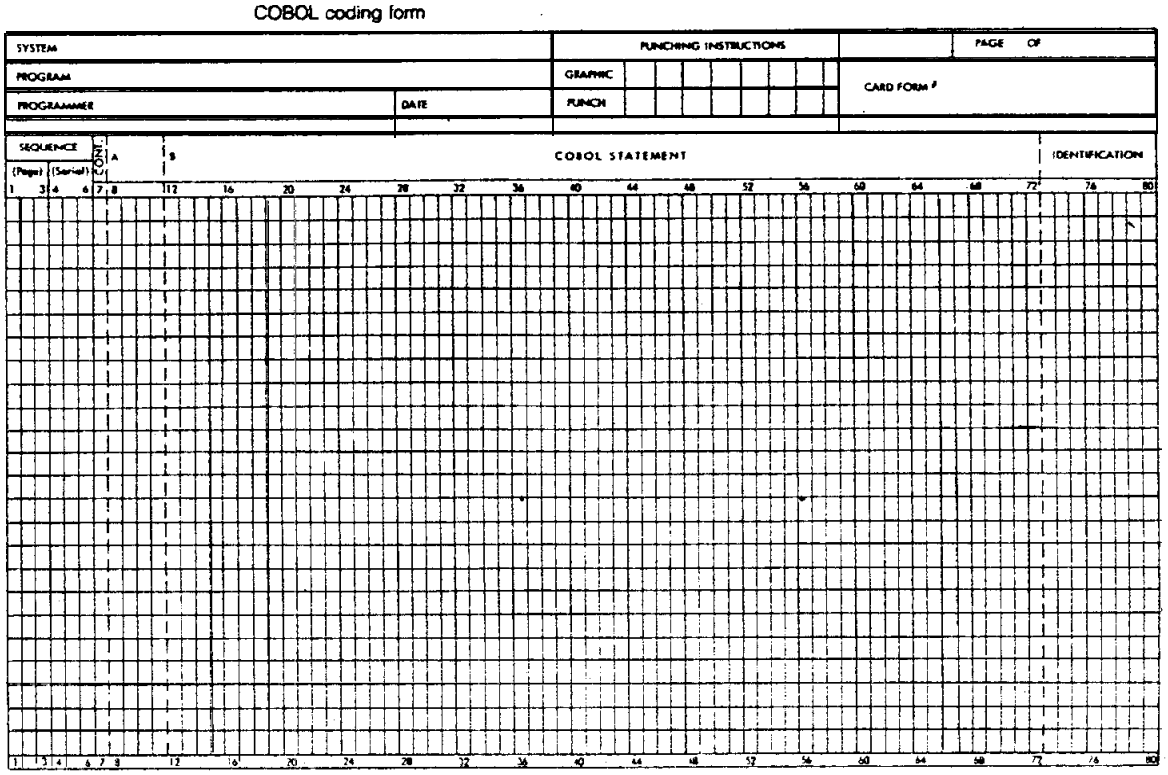
โปรแกรมภาษาโคบอลประกอบขึ้นจากส่วนต่าง ๆ คือ sentences, paragraphs, section และ divisions ซึ่งประกอบด้วย 4 divisions ดังต่อไปนี้คือ

1. Identification Division
2. Environment Division
3. Data Division
4. Procedure Division

นักพัฒนาภาษาโคบอลได้ออกแบบภาษาโคบอลให้มีโครงสร้างและมองดูเหมือนกับการเขียนรายงานภาษาอังกฤษ ดังนั้น โปรแกรมภาษาโคบอลจึงประกอบขึ้นจาก sentences, paragraphs, sections และ divisions โปรแกรมภาษาโคบอลแต่ละโปรแกรมต้องประกอบขึ้นด้วย 4 divisions จะขาด division ใด division หนึ่งไม่ได้ ในแต่ละ division อาจแบ่งเป็น section ต่าง ๆ ซึ่ง section จะประกอบด้วย paragraph ย่อย ๆ และทุกส่วนจะประกอบด้วย sentence ต่าง ๆ

COBOL coding form จะช่วยให้โปรแกรมเมอร์เขียนคำสั่งด้วยภาษาโคบอลให้เป็นไปตามกฎเกณฑ์ที่มีอยู่ ในบรรทัดหนึ่งมี 80 ตำแหน่งหรือคอลัมน์ คอลัมน์ที่ 1-3 ใช้สำหรับใส่เลขประจำหน้าของ coding form คอลัมน์ที่ 4-6 ใช้ใส่หมายเลขประจำบรรทัด ถ้าใส่เครื่องหมายดอกจัน (*) ในคอลัมน์ที่ 7 หมายความว่าเราจะเขียนคำอธิบายส่วนของโปรแกรม (Program documentation) หรือหมายเหตุ (remarks) คำสั่งในภาษาโคบอลเขียนในคอลัมน์ที่ 8-72 ในส่วนนี้ยังแบ่งออกเป็น 2 มาร์จิน (margin) คือมาร์จิน A เริ่มจากคอลัมน์ที่ 8 และมาร์จิน B เริ่มจากคอลัมน์ที่ 12 มาร์จิน A ใช้สำหรับเริ่มต้น division ใหม่ section ใหม่ หรือ paragraph

ใหม่ ส่วนมาร์จิน B ใช้สำหรับเริ่มต้น sentence ใหม่ คอลัมน์ที่ 73-80 ใช้สำหรับใส่ชื่อโปรแกรม ส่วนนี้คอมไพเลอร์จะไม่แปลเป็นภาษาเครื่อง



ตัวอย่างโปรแกรมซึ่งเขียนด้วยภาษาโคบอล

```
IDENTIFICATION    DIVISION.
PROGRAM-IO.     TOTAL-OF-15-NUMBERS.


ENVIRONMENT DIVISION.
CONFIGURATION   SECTION.
SOURCE-COMPUTER.    CYBER-74.
OBJECT-COMPUTER.    CVBER-74.
INPUT-OUTPUT   SECTION,
FILE-CONTROL.
     SELECT  DATA-FILE  ASSIGN  TO  CARD-READER-FZ
     SELECT  TOTAL-FILE  ASSIGN  TO  PRINTER-FZ.


DATA  DIVISION,
FILE  SECTION.
FD    DATA-FILE
      LABEL  RECORDS  ARE  OMITTED
      DATA  RECORD  IS  DATA-CARD.
01    DATA-CARD.
      02   DATA-VALUE      PICTURE IS 9(6)V99.
      02   FILLER          PICTURE  IS  X(72).
FD    TOTAL-FILE.
      LABEL  RECORDS  ARE  OMITTED.
      DATA  RECORD  IS  TOTAL-LINE.
01    TOTAL-LINE.
      02  TITLE           PICTURE IS  X(12)
                           VALUE  IS  "THE  TOTAL  IS".
      02   FILLER          PICTURE  IS  X(2).
      02   TOTAL-VALUE     PICTURE IS  9(13)V99.
      02   FILLER          PICTURE IS  X(102).
WORKING-STORAGE   SECTION.
77    RUNNING-TOTAL        PICTURE IS  9(13)V99
                           VALUE  IS  0.


PROCEDURE  DIVISION.
BEGIN-JOB.
     OPEN  INPUT  DATA-FILE
     OPEN  OUTPUT  TOTAL-FILE.
MAIN-LOOP.
     READ  DATA-FILE  AT  END  GO  TO  PRINT-ROUTINE.
     ADO  DATA-VALUE  TO  RUNNING-TOTAL,
     GO  TO  MAIN-LOOP.
PRINT-ROUTINE.
     MOVE  SPACES  TO  TOTAL-LINE.
     MOVE  RUNNING-TOTAL  TO  TOTAL-VALUE.
     WRITE  TOTAL-LINE.
END-OF-JOB.
     CLOSE  DATA-FILE.
     CLOSE  TOTAL-FILE.
     STOP  RUN.
```

บรรทัดแรกคือ IDENTIFICATION DIVISION เป็น sentence แรกในโปรแกรมภาษาโคบอล ซึ่งต้องตามด้วย PROGRAM-ID ซึ่งจะบอกชื่อของโปรแกรม ในที่นี้โปรแกรมนี้ชื่อ TOTAL-OF-15-NUMBERS ใน division นี้เราอาจเขียนชื่อผู้เขียนโปรแกรมหรือโปรแกรมเมอร์ วันที่เขียนโปรแกรมหรือหมายเหตุอื่น ๆ อีกได้ ใน ENVIRONMENT DIVISION นั้นใช้อธิบายเกี่ยวกับฮาร์ดแวร์ที่จะใช้ ส่วนนี้เท่านั้นที่ท่านต้องเปลี่ยนแปลงถ้าท่านเขียนโปรแกรมด้วย ANS COBOL แล้วจะนำไปใช้กับคอมพิวเตอร์ระบบอื่นได้ division นี้ประกอบด้วย 2 sections คือ CONFIGURATION SECTION และ INPUT-OUTPUT section ใน paragraph FILE-CONTROL ประกอบขึ้นด้วย 2 SELECT sentences ใน DATA DIVISION ใช้อธิบายแฟ้มข้อมูลเข้า/ออกที่ใช้ทั้งหมดไว้ใน FILE SECTION และแฟ้มข้อมูลที่ใช้ในการทำงาน (working file) ซึ่งใช้เก็บข้อมูลจากขั้นของการประมวลผลจะถูกอธิบายใน WORKING-STORAGE SECTION ใน PROCEDURE DIVISION จะอธิบายวิธีจัดกระทำกับข้อมูลเข้าเพื่อให้ได้ข้อมูลออกตามความต้องการ จากตัวอย่างจะเห็นว่า division นี้ประกอบด้วย 4 paragraph ชื่อ BEGIN-JOB, MAIN-LOOP, PRINT-ROUTINE และ END-OF-JOB และแต่ละ paragraph ประกอบด้วยคำสั่งต่าง ๆ

2.3.4 ภาษาปาสกาล (Pascal)

ชื่อของภาษาปาสกาลนั้นตั้งตามชื่อของนักคณิตศาสตร์ชาวฝรั่งเศสผู้มีชื่อเสียง คือ แบลส์ ปาสกาล (Blaise Pascal) ภาษานี้เป็นภาษาที่ค่อนข้างใหม่แต่มีอนาคตสดใสมาก เป็นภาษาที่ใช้กับระบบเรียลไทม เป็นภาษาชนิด general purpose, procedure oriented ซึ่งใช้งานทั้งทางธุรกิจและการคิดคำนวณทางวิทยาศาสตร์ ผู้คิดพัฒนาภาษานี้คือ นิคลอส เวิร์ธ (Niklaus Wirth) ซึ่งทำงานอยู่ที่สถาบันเทคโนโลยีของรัฐในเมืองซูริค (Zurick) ประเทศสวิตเซอร์แลนด์ในปี ค.ศ.1968 ภาษานี้เป็นภาษาที่เรียนรู้ได้ค่อนข้างง่ายและใช้เขียนโปรแกรมแบบโครงสร้างได้ ภาษานี้เหมาะกับเครื่องมินิคอมพิวเตอร์และไมโครคอมพิวเตอร์ และบริษัทผู้ผลิตเครื่องคอมพิวเตอร์ทั้ง 2 ระบบ ก็ได้ลงทุนกันเป็นหลาย ๆ ล้านเหรียญสหรัฐเพื่อพัฒนาคอมไพเลอร์ของภาษานี้ ภาษาปาสกาลใช้กับระบบคอมพิวเตอร์ระบบใหญ่ ๆ ด้วย ภาษานี้นิยมสอนกันในวิทยาลัยและมหาวิทยาลัยหลายแห่งในยุโรปและอเมริกา

ตัวอย่างโปรแกรมซึ่งเขียนด้วยภาษาปาสกาล

```
PROGRAM SUM15 (INPUT,OUTPUT);
(* *)
(* DECLARE VARIABLES *)
    VAR  X : REAL;
         TOTAL : REAL;
         I = INTEGER;
    BEGIN
(* *)
(* INITIALIZE ACCUMULATOR *)
         TOTAL := 0;
         FOR I :=1 TO 15 DO
         BEGIN
              READ(X);
              TOTAL := TOTAL + X
         END;
         WRITELN;
         WRITELN('THE TOTAL IS ',TOTAL)
    END.
```

โปรแกรมที่เขียนด้วยภาษาปาสกาลข้างต้น จะหาผลบวกของเลข 15 จำนวน โปรแกรมภาษาปาสกาลคล้ายคลึงกับโปรแกรมในภาษาเบสิกและภาษาฟอร์แทรน ในบรรทัดแรกเป็นชื่อของโปรแกรม คือ SUM15 มีการกำหนดตัวแปรที่จะใช้ในโปรแกรมและชนิดของมัน จะเห็นว่าตัวแปร X และ TOTAL ถูกกำหนดให้เป็นตัวแปรจำนวนจริง (real variable) ซึ่งจะเก็บเลขจำนวนจริง ส่วนตัวแปร I เป็นตัวแปรจำนวนเต็ม (integer variable) ซึ่งจะเก็บเลขจำนวนเต็ม ส่วนสำคัญของคำสั่งเริ่มจากคำว่า BEGIN และจบด้วยคำว่า END ในการเขียนคำสั่งในภาษาปาสกาลไม่กำหนดคอลัมน์แน่นอน เช่น ภาษาฟอร์แทรนและภาษาโคบอล เราอาจย่อหน้าคำสั่งเพื่อทำให้โปรแกรมง่ายต่อการอ่านและการเข้าใจ ให้สังเกตบรรทัดที่ 3 บรรทัดนั้นเป็น comment statement ซึ่งเราใช้เพื่ออธิบายส่วนของโปรแกรมหรือเขียนหมายเหตุตามต้องการ เราอาจเขียนคำอธิบายหรือข้อความใด ๆ ระหว่างเครื่องหมาย (* และ *)

2.3.5 ภาษาเอดา (Ada)

ภาษาอีกภาษาหนึ่งซึ่งเป็นภาษาใหม่และจะมีอนาคตสดใสเช่นกัน คือ ภาษาเอดา ชื่อของภาษาตั้งตามชื่อของ เอดา ออกัสตา (Ada Augusta) ซึ่งเป็นเคาน์เตสของ Lovelace เอดาเป็นเพื่อนสนิทของชาลส์ แบบเบจ ซึ่งเป็นศาสตราจารย์ทางคณิตศาสตร์ของมหาวิทยาลัยแคมบริดจ์ เอดาทำงานกับชาลส์ แบบเบจ โดยช่วยคิดออกแบบเครื่องวิเคราะห์ (Analytical engine) ในช่วงครึ่งแรกของศตวรรษที่19 เอดาได้รับการยกย่องว่าเป็นโปรแกรมเมอร์คนแรก

ภาษาเอดา ถูกพัฒนาขึ้นตามความต้องการและสนับสนุนของกระทรวงกลาโหมสหรัฐ (U.S. Department of Defense) เพื่อใช้ในงานทางทหาร ภาษานี้ถูกนำออกมาสู่วงการในปลายปี ค.ศ.1980 หลังจากถูกทดสอบและถูกประเมินผลก็เป็นที่คาดหมายว่าในอนาคต

โปรแกรมส่วนใหญ่ของกระทรวงกลาโหมสหรัฐจะเขียนด้วยภาษานี้ นอกจากนั้นภาษานี้ได้เป็นที่นิยมในวงการอุตสาหกรรม มหาวิทยาลัย และสาขาต่าง ๆ ที่สำคัญของทางทหาร

ตัวอย่างโปรแกรมซึ่งเขียนด้วยภาษาเอดา

```
procedure SIMPLE_TOTAL is
X, I, TOTAL : INTEGER
TOTAL : = 0;
for I in 1 . . 15 loop
   TOTAL : = TOTAL + X(I);
end loop;
PUT (TOTAL);
end SIMPLE_TOTAL
```

โปรแกรมตัวอย่างข้างต้นจะหาผลบวกของเลข 15 จำนวน โปรแกรมนี้มีส่วนคล้ายกับภาษาปาสกาลมาก คำสั่งแรกเป็นคำสั่ง procedure คำสั่งนี้ใช้กำหนดชื่อของโปรแกรม ซึ่งคือ SIMPLE_TOTAL ถ้าเปรียบเทียบโปรแกรมตัวอย่างนี้กับโปรแกรมในภาษาอื่น ๆ เท่าที่ผ่านมา จะเห็นว่ามีส่วนที่คล้ายคลึงกันกับภาษาอื่น ๆ ในระดับสูงมาก แต่อย่างไรก็ดีภาษาเอดามีลักษณะพิเศษอีกหลายอย่างซึ่งไม่ได้แสดงในโปรแกรมง่าย ๆ โปรแกรมนี้

2.3.6 ภาษาอาร์พีจี (RPG)

RPG ย่อมาจาก Report Program Generator

ภาษาอาร์พีจีเป็นภาษา business-oriented และ general purpose เป็น problem-oriented language ด้วย เป็นภาษาที่ใช้ในระบบเป็นกลุ่มกับระบบคอมพิวเตอร์ เช่น IBM system/3, IBM system/34 ในการเขียนคำสั่งโปรแกรมเมอร์จะต้องใช้ coding form 4 ฟอร์ม หรือที่เรียกว่า specification sheet แต่ละฟอร์มจะใช้เพื่อวัตถุประสงค์ที่ต่างกัน ภาษานี้พัฒนาจากภาษาอาร์พีจี เป็นอาร์พีจีสอง (RPG II) และอาร์พีจีสาม (RPG III) อาร์พีจีสามสามารถใช้กับระบบอินเทอร์แอคทีฟ (interactive mode คือระบบที่มีการโต้ตอบกันระหว่างเครื่องคอมพิวเตอร์กับผู้ใช้งาน) ภาษานี้ไม่เหมาะกับการคำนวณทางคณิตศาสตร์

รูปต่อไปคือ specification sheet ทั้ง 4 ฟอร์ม ของภาษาอาร์พีจี (มี 5 รูป เพราะรูปที่ 1 และ 2 ถือว่าเป็นฟอร์มเดียว)

IBM

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic								
	Punch								

Page 1 2

Program Identification 75 76 77 78 79 80

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	Debug	MFCM Stacking Sequence	Sterling: Input-Shillings	Sterling: Input-Pence	Sterling: Output-Shillings	Sterling: Output-Pence	Inverted Print	360/20 2501 Buffer	Number Of Print Positions	Alternate Collating Sequence	Model 20: Address to Start	Model 20: Work Tapes	Model 20: Overlay Open	Model 20: Overlap Printer	Model 20: Binary Search	Model 20: Tape Error	Model 20: 2152 Checking	Model 20: Inquiry	Model 20: Read/Write/Compute	Model 20: Keyboard Output	Model 20: Sign Handling	Model 20: 1P Forms Position	Model 20: Indicator Setting	Model 20: File Translation	Model 20: Punch MFCU Zeros	Model 20: Nonprint Characters	Table-Load Halt	Shared I/O	Field Print	Formatted Core Dump	RPG to RPG II Conversion	Refer to the specific System Reference Library manual for actual entries.
3 4 5	6	7 8 9	10	11	12 13 14	15	16	17	18	19	20	21	22	23 24 25	26	27 28 29 30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 1	H																																			

File Description Specifications

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	File Designation: E	End of File: A/D	Sequence	File Format: F/V/S/M/D	Block Length	Record Length	L/R	Length of Key Field or of Record Address Field	A/P/I/K	Record Address Type	I/O/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Core Index	Continuation Lines: K	Option	Entry	A/U	File Addition/Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	R/U/N	Tape Rewind	File Condition U1-U8
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19	20 21 22 23	24 25 26 27	28	29 30 31	32	33 34	35 36 37 38	39	40 41 42 43 44 45 46	47 48 49 50 51 52	53	54 55 56 57 58 59	60 61 62 63 64 65	66	67 68 69	70	71 72	73 74
0 2	F																																			
0 3	F																																			
0 4	F																																			
0 5	F																																			
0 6	F																																			
0 7	F																																			
	F																																			
	F																																			

*No. of forms per pad may vary slightly.

IBM

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page (1 2)

Program Identification (75 76 77 78 79 80)

Extension Specifications

Line (3 4 5)	Form Type (6)	Record Sequence of the Chaining File / Number of the Chaining Field / From Filename (7 8 9 10 11 12 13 14 15 16 17 18)	To Filename (19 20 21 22 23 24 25 26)	Table or Array Name (27 28 29 30 31 32)	Number of Entries Per Record (33 34 35)	Number of Entries Per Table or Array (36 37 38 39)	Length of Entry (40 41 42)	P = Packed/B = Binary (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46 47 48 49 50 51)	Length of Entry (52 53 54)	P = Packed/B = Binary (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74)
0 1	E															
0 2	E															
0 3	E															
0 4	E															
0 5	E															
0 6	E															
0 7	E															
0 8	E															
0 9	E															
1 0	E															

Line Counter Specifications

Line (3 4 5)	Form Type (6)	Filename (7 8 9 10 11 12 13 14)	1 Line Number (15 16 17)	1 FL or Channel Number (18 19)	2 Line Number (20 21 22)	2 OL or Channel Number (23 24)	3 Line Number (25 26 27)	3 Channel Number (28 29)	4 Line Number (30 31 32)	4 Channel Number (33 34)	5 Line Number (35 36 37)	5 Channel Number (38 39)	6 Line Number (40 41 42)	6 Channel Number (43 44)	7 Line Number (45 46 47)	7 Channel Number (48 49)	8 Line Number (50 51 52)	8 Channel Number (53 54)	9 Line Number (55 56 57)	9 Channel Number (58 59)	10 Line Number (60 61 62)	10 Channel Number (63 64)	11 Line Number (65 66 67)	11 Channel Number (68 69)	12 Line Number (70 71 72)	12 Channel Number (73 74)
1 1	L																									
1 2	L																									
1 3	L																									

IBM

International Business Machines Corporation

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 1 2

Program Identification 75 76 77 78 79 80

Line 3 4 5	Form Type 6	Filename 7 8 9 10 11 12 13 14	Sequence 15 16	Number (1-N) 17	Option (O) 18	Record Identifying Indicator or ** 19 20	Record Identification Codes 1: Position 21 22 23 24	1: Not (N) 25	1: C/Z/D 26	1: Character 27	2: Position 28 29 30 31	2: Not (N) 32	2: C/Z/D 33	2: Character 34	3: Position 35 36 37 38	3: Not (N) 39	3: C/Z/D 40	3: Character 41	Stacker Select 42	P = Packed/B = Binary 43	Field Location: From 44 45 46 47	Field Location: To 48 49 50 51	Decimal Positions 52	Field Name 53 54 55 56 57 58	Control Level (L1-L9) 59 60	Matching Fields or Chaining Fields 61 62	Field Record Relation 63 64	Field Indicators: Plus 65 66	Field Indicators: Minus 67 68	Field Indicators: Zero or Blank 69 70	Sterling Sign Position 71 72 73 74
0 1	I																														
0 2	I																														
0 3	I																														
0 4	I																														
0 5	I																														
0 6	I																														
0 7	I																														
0 8	I																														
0 9	I																														
1 0	I																														
1 1	I																														
1 2	I																														
1 3	I																														
1 4	I																														
1 5	I																														
	I																														
	I																														
	I																														
	I																														
	I																														

*Number of forms per pad may vary slightly

OR 213

IBM

International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (High 1>2 / Low 1<2 / Equal 1=2); Lookup (Table (Factor 2) is High / Low / Equal)	Comments
3 4 5	6	7 8	9 10 11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53	54 55 56 57 58 59	60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
01	C											
02	C											
03	C											
04	C											
05	C											
06	C											
07	C											
08	C											
09	C											
10	C											
11	C											
12	C											
13	C											
14	C											
15	C											
	C											
	C											
	C											
	C											
	C											

505

IBM

International Business Machines Corporation

RPG OUTPUT - FORMAT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select/Fetch Overflow (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Output Indicators And Not	Output Indicators Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	P = Packed/B = Binary	Constant or Edit Word	Sterling Sign Position
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19 20	21 22	23 24 25	26 27 28	29 30 31	32 38 34 35 36 37	38	39	40 41 42 43	44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70	71 72 73 74
0 1	O																	
0 2	O																	
0 3	O																	
0 4	O																	
0 5	O																	
0 6	O																	
0 7	O																	
0 8	O																	
0 9	O																	
1 0	O																	
1 1	O																	
1 2	O																	
1 3	O																	
1 4	O																	
1 5	O																	
	O																	
	O																	
	O																	
	O																	
	O																	

Edit Codes

Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

*Number of forms per pad may vary slightly

2.3.7 ภาษาพีแอลวัน (PL/I)

PL/I ย่อมาจาก Programming Language I

ประมาณต้นปี ค.ศ.1960 บริษัท IBM และคณะกรรมการผู้ใช้ระบบคอมพิวเตอร์ IBM system/360 ได้พัฒนาภาษาขึ้นเพื่อให้เป็นภาษาที่ใช้ได้กับงานทั่ว ๆ ไปหรือเป็น universal langauge ในกลางปี ค.ศ.1960 จึงได้ภาษาพีแอลวันออกมาสู่วงการ โดยนำเอาส่วนดีของภาษาฟอร์แทรนและภาษาโคบอลมารวมกัน ภาษานี้มีโครงสร้างของ division คล้ายกับภาษาโคบอล ภาษานี้เป็นภาษา general purpose และ procedure oriented ภาษานี้สามารถใช้ได้กับงานทางธุรกิจและงานทางวิทยาศาสตร์ ภาษานี้ต้องใช้หน่วยความจำหลักขนาดใหญ่และไม่เป็นภาษามาตรฐานเหมือนกับภาษาฟอร์แทรนและภาษาโคบอล เป็นภาษาที่ไม่ค่อยแพร่หลายนัก ส่วนใหญ่ต้องใช้กับเครื่องของบริษัท IBM ในปี ค.ศ.1976 สถาบันมาตรฐานแห่งชาติของสหรัฐอเมริกา (ANSI) ได้สร้างภาษาพีแอลวันมาตรฐานขึ้นมา และส่วนหนึ่งของภาษาคือ "subset G" นั้นสามารถใช้กับคอมพิวเตอร์ขนาดเล็กได้ด้วย เนื่องจากข้อจำกัดของภาษามีมาก จึงทำให้ยากกว่าการเรียนภาษาฟอร์แทรนหรือภาษาโคบอลโดยตรง ในการเขียนไม่ต้องใช้ coding form

ตัวอย่างโปรแกรมซึ่งเขียนด้วยภาษาพีแอลวัน

```
PL/1 OPTIMIZING COMPILER      WATER: PROCEDURE OPTIONS (MAIN);

                              SOURCE LISTING

STMT LEV NT

   1    0 WATER: PROCEDURE OPTIONS (MAIN);
   2  1 0       DECLARE SYSIN INPUT. SYSPRINT OUTPUT;
   3  1 0       DECLARE NAME CHARACTER (32);
   4  1 0       DECLARE KWH FIXED DECIMAL (4);
   5  1 0       DECLARE K FIXED DECIMAL (8.2);
   6  1 0       DECLARE N FIXED DECIMAL (6.2);
   7  1 0       ON ENDFILE(SYSIN) STOP;
                /* PRINT HEADER ● /
   8  1 0       PUT PAGE;
   9  1 0       PUT SKIP (4) EDIT
                ('NAME','USED(KWH)','AMOUNT(CENT)','PAY(DOLLAR)'
                (X(26),A,X(26),A,X,(12),A,X(12),A);
  10  1 0 QUANT: GET LIST (NAME.KWH),
  11  1 0       IF KWH<=90 THEN K=KWH*. 11.75; ELSE
  12  1 0       IF (KWH<90) & (KWN<=540) THEN K=90*11.75+(KWH-90)*3.74; ELSE
  13  1 0       K=90*11.75+450*3.74+(KWH-540)*3.04;
  14  1 0       N=K/100;
  15  1 0       PUT SKIP(2) EDIT (N4ME.KWH.K.N)
                        (X(12),A,X(12),F(4),X(17),F(8,2),X(16),F(6.2));
  16  1 0       GO TO QUANT;
  17  1 0       END WATER;
```

2.3.8 ภาษาเอพีแอล (APL)

APL ย่อมาจาก A Programming Language

ภาษาเอพีแอลได้รับการพัฒนาขึ้นในปี ค.ศ.1962 เป็นภาษา General purpose และ procedure-oriented เป็นภาษาที่ใช้กับระบบเรียลไทม ซึ่งเริ่มแรกนั้นพัฒนาเพื่อการประยุกต์ใช้ทางวิทยาศาสตร์ ภาษานี้มีโครงสร้างแบบ free-form และใช้ตัวกระทำพิเศษซึ่งมีรูปร่างคล้ายกับอักษรกรีกในการทำงานที่ยุ่งยากซับซ้อน เนื่องจากตัวกระทำพิเศษเหล่านี้จะต้องใช้แป้นพิมพ์พิเศษในการเขียนโปรแกรมด้วยภาษานี้ ภาษานี้ใช้ได้กับระบบคอมพิวเตอร์ทั้งขนาดเล็กและขนาดใหญ่ คอมพิวเตอร์ขนาดเล็กของบริษัท IBM ขนาดตั้งโต๊ะคือ IBM 5110 สามารถสับสวิทช์เปลี่ยน mode ไปมาระหว่าง BASIC mode และ APL mode ได้ เมื่อต้องการเขียนโปรแกรมในภาษาเบสิกหรือภาษาเอพีแอลตามต้องการ ภาษานี้ยังไม่เป็นที่นิยมเท่าภาษาเบสิก แต่ในอนาคตจะใช้มากขึ้น

2.3.9 ภาษาอัลกอล (ALGOL)

ALGOL ย่อมาจาก Algorithmic Language

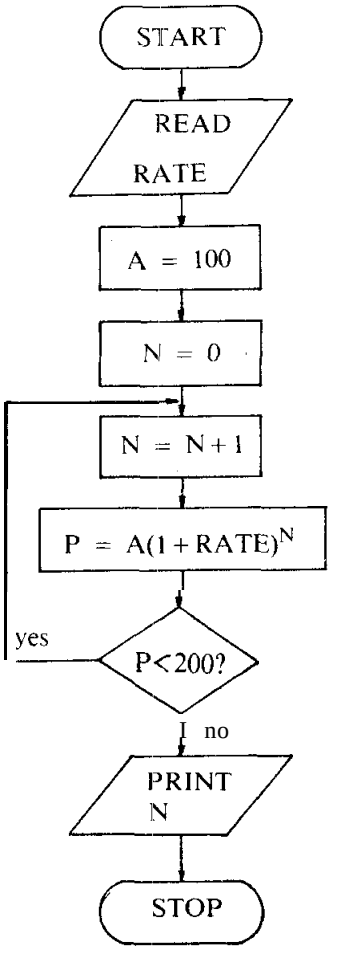
ภาษาอัลกอลเป็นภาษาชนิด special purpose ซึ่งเหมาะกับงานทางด้านวิทยาศาสตร์ มีความสามารถมากกว่าและเขียนง่ายกว่าเมื่อเทียบกับภาษาฟอร์แทรน ผู้พัฒนาภาษานี้นั้นคือกลุ่มของนักคณิตศาสตร์ชาติต่าง ๆ โดยมีจุดมุ่งหมายจะให้เป็นภาษาที่ใช้แทนภาษาฟอร์แทรน แต่ปรากฏว่าไม่ได้รับความนิยมในวงการธุรกิจอุตสาหกรรมเลย ภาษานี้ยังใช้อยู่ภายในมหาวิทยาลัยในยุโรปเท่านั้น

ภาษาคอมพิวเตอร์เท่าที่กล่าวมาใน 6.2.3.1-6.2.3.8 นั้นเป็นภาษาชนิด general purpose ทั้งสิ้น โปรแกรมที่เขียนขึ้นด้วยภาษาเหล่านั้นสามารถแก้ปัญหานานาชนิดได้ตั้งแต่ปัญหาทางธุรกิจ ทางวิทยาศาสตร์ และในสาขาอื่น ๆ นอกจากภาษาชนิด general purpose แล้ว ยังมีภาษาชนิด special purpose ซึ่งจะปฏิบัติงานหลักเพียง 1 งาน หรือทำหน้าที่เพียง 1 อย่าง ภาษาชนิดดังกล่าวจะทำงานหลักของมันอย่างมีประสิทธิภาพสูง ภาษาเหล่านี้ถูกออกแบบขึ้นเพื่อให้ใช้ได้ง่าย ภาษาชนิดนี้มีอยู่หลายร้อยภาษา เช่น ภาษาอัลกอล สำหรับงานทางวิทยาศาสตร์ ถ้าสำหรับงานประมวลผลด้านแถวของตัวอักขระ (string processing) เช่น ภาษา LISP (List Processing Language) ภาษา SNOBOL (String Oriented Symbolic Language) สำหรับงานด้านเตรียมโปรแกรมการสอน (educational program) เช่น ภาษา PILOT (Programmed Inquiry Learning Or Teaching) และสำหรับงานทางโรงงาน เช่น ภาษา APT (Automatically Programmed Tooling)

นอกจากภาษาดังกล่าวแล้วบริษัทผู้ผลิตคอมพิวเตอร์พยายามพัฒนาภาษาใหม่ ๆ ขึ้นมาโดยมีวัตถุประสงค์ให้วิธีการเขียนโปรแกรมง่ายสำหรับผู้ใช้ทั่ว ๆ ไป ผู้เชี่ยวชาญบางท่านเรียกภาษาเหล่านี้ว่า เป็นภาษายุคที่ 4 (fourth generation language) โดยเรียกว่า Data Base Language หรือ Query language ทั้งนี้ การนับยุคของภาษานั้นคือยุคที่ 1 คือภาษาเครื่อง ยุคที่ 2 คือ ภาษาในขั้นต่ำ และยุคที่ 3 คือภาษาในขั้นสูง ที่เรียกว่าภาษาในยุคที่ 4 นี้ เนื่องจากความสามารถของมันเหนือภาษาในระดับสูง ตัวอย่างเช่น ภาษา Mapper ซึ่งพัฒนาโดยบริษัท Sperry Univac ทางบริษัทกล่าวไว้ว่า ภายใน 2-3 วัน ของการฝึกฝน นักบริหารส่วนใหญ่จะสามารถพัฒนารายงานของตนเองจากระบบคอมพิวเตอร์ได้ ด้วยการใช้คำสั่งและวิธีการที่ง่ายต่อการเข้าใจเพื่อผลิตรายงานที่ซับซ้อนจากฐานข้อมูล (Data Base) ที่มีอยู่ได้ ภาษาอื่น ๆ ที่คล้ายภาษา Mapper ซึ่งเรียกว่าเป็นภาษา User friendly เช่น ภาษา LOGO ซึ่งได้รับการพัฒนาในช่วงปี ค.ศ.1970-1972 และใช้กับคอมพิวเตอร์ส่วนตัว ภาษา FOCUS ภาษา FIRST เป็นต้น

3. ตัวอย่างของการใช้ผังโปรแกรมช่วยการเขียนโปรแกรม

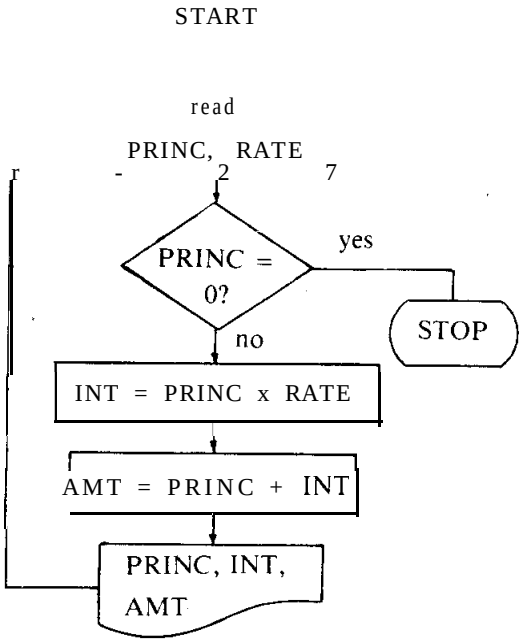
ตัวอย่างที่ 1



โปรแกรมภาษาฟอร์แทรน

```
C       PROGRAM IN FORTRAN LANGUAGE
        READ (5,2) RATE
2       FORMAT (F5.2)
        A = 100
        N = 0
12      N = N+1
        P = A*(1.0+RATE)**N
        IF (P.LT.200) GO TO 12
        WRITE (6,3) N
3       FORMAT (13)
        STOP
        END
```

ตัวอย่างที่ 2



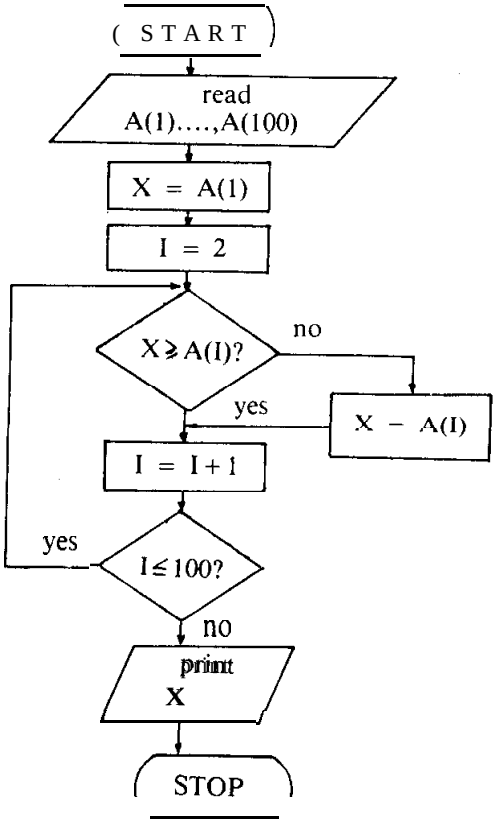
โปรแกรมภาษาพีแอลวัน

```
STRT: GET LIST (PRINC, RATE);
IF PRINC = 0 THEN STOP;
INT = PRINC * RATE;
AMT = PRINC + INT;
PUT LIST (PRINC, INT, AMT);
GO TO' STRT:
```

โปรแกรมภาษาฟอร์แทรน

```
13  READ (5, 10) PRINC, RATE
10  FORMAT (F 6.0, F 5.2)
    IF (PRINC. EC?. 0) STOP
    INT = PRINC*RATE
    AMT = PRINC + INT
    WRITE (6, 11) PRINC, INT, AMT
11  FORMAT (5X, F 7.0, 2X, F 5.2, F 9.2)
    GOT0 13
    END
```

ตัวอย่างที่ 3



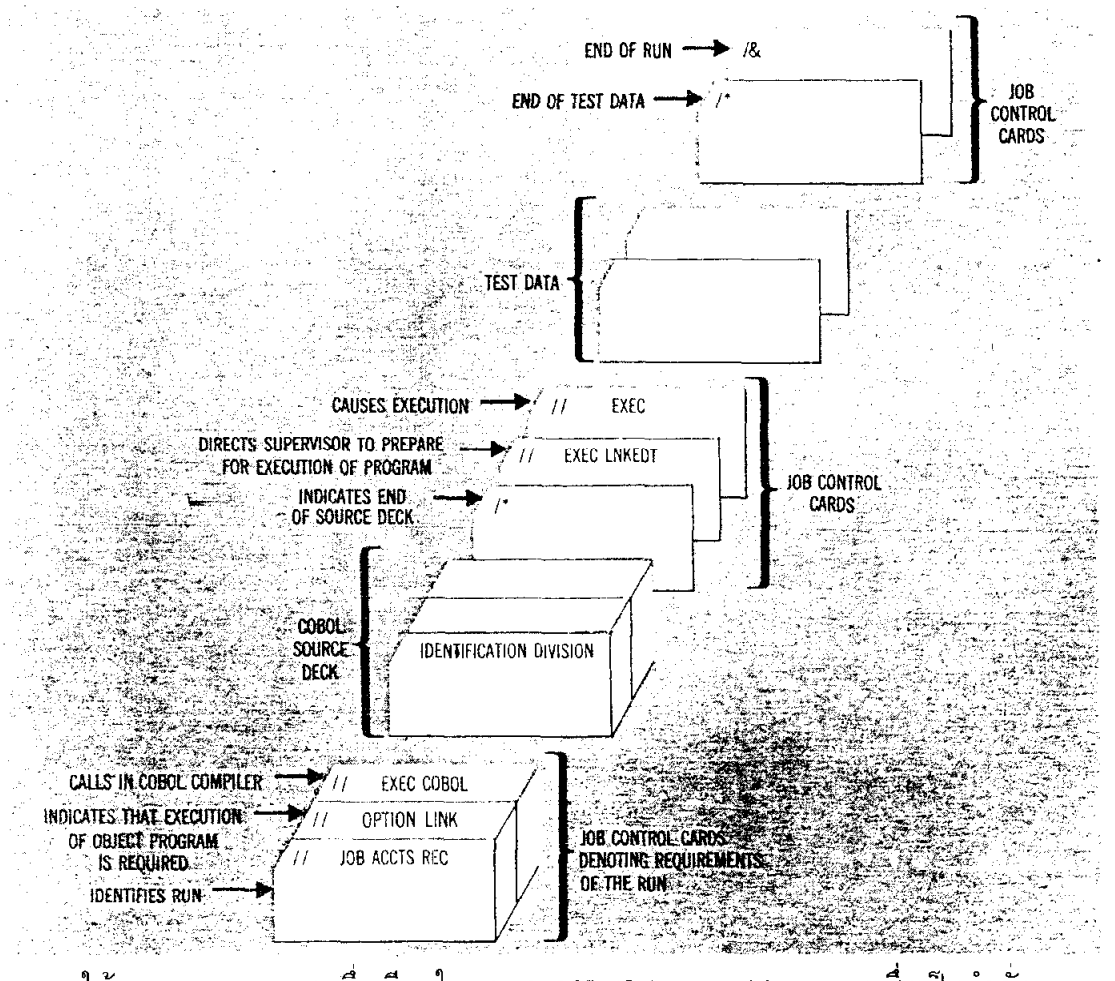
โปรแกรมภาษาเบสิก

```
090 REM BASIC EXAMPLE
100 DIM A (100)
110 FOR I = 1 TO 100
120 READ A (I)
130 NEXT I
140 LETX = A(1)
150 FOR I = 2 TO 100
160 IF X >= A(1) THEN 180
170 LETX = A(I)
180 NEXT I
190 PRINT "THE LARGEST";
200 PRINT "ELEMENT IN THE".
210 PRINT "ARRAY = "; X
220 DATA 20, 16, 16,.....
230 DATA 47, 32, 64, .....
240 END
RUN
```

โปรแกรมภาษาฟอร์แทรน (สำหรับผังโปรแกรมในตัวอย่างที 3)

คอลัมน์ 1 2 3 4 5 6 7

```
      DIMENSION A (100)
      READ (5,10) (A (I), I = 1, 100)
10    FORMAT (10F2.0)
      X = A(1)
      DO 12 I = 2, 100
      IF (X .GE.A(I)) GO TO 12
      X = A(I)
12    CONTINUE
      WRITE (6,9) X
9     FORMAT (1X,'THE LARGEST ELEMENT IN THE ARRAY = ',F3.0)
      STOP
      END
```



การใช้ Job control cards ซึ่งเขียนในภาษา JCL (Job control language) ซึ่งเป็นคำสั่ง (Command) ให้โปรแกรมควบคุมระบบควบคุมการแปลและปฏิบัติตามโปรแกรม

๔.การเปรียบเทียบภาษาที่ใช้ในการเขียนโปรแกรม

เนื่องจากระบบคอมพิวเตอร์ส่วนมากสามารถรับคำสั่งที่เขียนด้วยภาษาในระดับสูงได้หลายภาษา ดังนั้น จึงมีความสำคัญในการที่จะเลือกภาษาที่ดีที่สุดสำหรับการประยุกต์ใช้เฉพาะอย่าง ดังนั้น จึงจะเปรียบเทียบภาษาในระดับสูงบางภาษา ดังนี้

ภาษาเบสิกนั้นเรียนรู้ง่าย ใช้กับระบบอินเทอร์แอคทีฟโดยธรรมชาติของมัน ภาษานี้ใช้ได้กับระบบคอมพิวเตอร์จำนวนมาก ภาษานี้ใช้ได้กับการคำนวณและการจัดการกับแถวของตัวอักขระ (character string) ไม่ต้องใช้เทอร์มินัลและคีย์บอร์ดพิเศษแต่อย่างใด เนื่องจากมันไม่ต้องการใช้หน่วยความจำขนาดใหญ่ภาษาเบสิกจึงใช้ได้กับไมโครคอมพิวเตอร์ คอมพิวเตอร์ขนาดเล็ก กลาง และใหญ่ด้วย อย่างไรก็ดีภาษานี้ไม่เป็นภาษามาตรฐานอย่างสมบูรณ์ โปรแกรมในภาษานี้เมื่อใช้กับคอมพิวเตอร์ระบบหนึ่งแล้วจะต้องถูกปรับปรุงแก้ไขก่อนนำไปใช้กับคอมพิวเตอร์ระบบอื่น ๆ ภาษานี้มีประสิทธิภาพไม่สูงมาก ไม่เป็นเอกสารในตัวของมันเอง (not self-documenting) และไม่มีข้อความคล้ายภาษาอังกฤษ (English-like)

ภาษาฟอร์แทรนมีความสามารถในการคำนวณเป็นเลิศ มีใช้กับระบบคอมพิวเตอร์จำนวนมากและเป็นภาษาที่รัดกุมกะทัดรัด ภาษานี้ไม่ต้องใช้หน่วยความจำหลักขนาดใหญ่ ภาษาฟอร์แทรน 77 เท่านั้นที่จัดการกับแถวของตัวอักขระได้ภาษานี้ไม่เป็นเอกสารในตัวของมันเองและไม่มีข้อความคล้ายภาษาอังกฤษ และมีกฎเกณฑ์หลายอย่างที่ต้องทำตามอย่างเคร่งครัดเพื่อหลีกเลี่ยงข้อผิดพลาด

ภาษาโคบอลเป็นภาษาที่มีลักษณะเป็นข้อความคล้ายภาษาอังกฤษ เป็นเอกสารในตัวของมันเอง เป็นภาษามาตรฐานและง่ายต่อการเรียนรู้และทำความเข้าใจ เป็นภาษาที่นิยมมากสำหรับการประยุกต์ใช้ทางธุรกิจ เนื่องจากโปรแกรมภาษาโคบอลนั้นเป็นข้อความคล้ายภาษาอังกฤษจึงทำให้ง่ายต่อการทดสอบและแก้ไขกว่าภาษาอื่น ๆ ภาษานี้มีใช้กับระบบคอมพิวเตอร์จำนวนมาก และเนื่องจากมันเป็นภาษามาตรฐานมันจึงง่ายต่อการแก้ไขดัดแปลงเพื่อใช้กับคอมพิวเตอร์ระบบอื่น นั่นคือทำการแก้ไขเพียง Environment Division เท่านั้น อย่างไรก็ดีภาษานี้ต้องระวังการใช้กริยาตามหลักไวยากรณ์อังกฤษด้วย ในบางกรณีเราอาจจะต้องเขียนโปรแกรมยาวเป็น 2-3 เท่าของโปรแกรมที่เขียนด้วยภาษาอื่น สำหรับปัญหาเดียวกัน ความสามารถในการคำนวณของภาษานี้ไม่ดี เป็นภาษาที่ต้องใช้หน่วยความจำหลักขนาดใหญ่ เรามักไม่พบภาษาโคบอลที่ใช้กับไมโครคอมพิวเตอร์

ภาษาปาสกาลและภาษาเอดาง่ายต่อการเรียนรู้ และโดยธรรมชาติของมันใช้กับระบบ

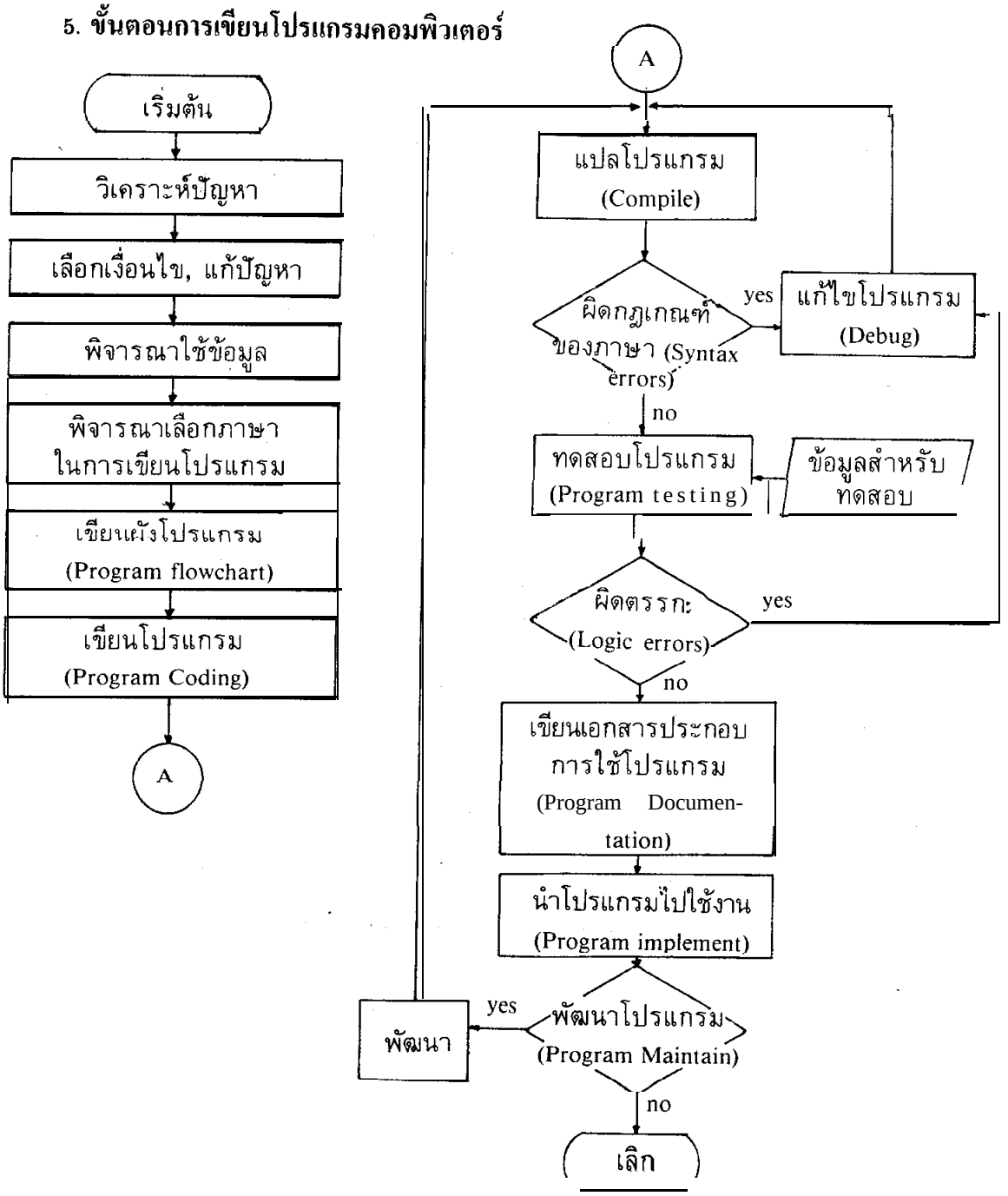
อินเทอร์แอคทีฟและเป็นภาษาที่ใช้เขียนโปรแกรมแบบโครงสร้างได้ ภาษาเหล่านี้มี default option จำนวนมากที่ทำให้โปรแกรมเมอร์อาจเขียนโปรแกรมง่าย ๆ หรือที่ยุ่งยากซับซ้อนได้ ภาษาเหล่านี้ไม่เป็นเอกสารในตัวเอง ไม่มีข้อความคล้ายภาษาอังกฤษ และไม่เป็นมาตรฐานอย่างสมบูรณ์

ภาษาอาร์พีจี เป็นภาษาที่เป็น problem-oriented และใช้ได้กับระบบคอมพิวเตอร์ขนาดเล็กและขนาดกลางจำนวนมาก ไม่ต้องใช้หน่วยความจำขนาดใหญ่และการเรียนรู้ค่อนข้างง่าย เป็นภาษาที่ดีสำหรับการผลิตรายงานและเอกสารสำหรับธุรกิจ ภาษานี้ไม่เหมาะกับการคำนวณ ดังนั้นจึงไม่เป็นภาษาที่ดีสำหรับงานทางด้านวิทยาศาสตร์ ภาษานี้ไม่เป็นภาษามาตรฐาน หรือเป็นภาษาที่มีข้อความคล้ายภาษาอังกฤษ

ภาษาพีแอลวัน มีความสามารถทางการคำนวณสูงและมีบิลท์อินฟังก์ชั่นจำนวนมาก ที่ทำให้การเขียนโปรแกรมง่ายขึ้น มี default option จำนวนมาก ที่ทำให้ขบวนการการเขียนโปร-แกรมง่าย ภาษานี้สามารถจัดการกับขบวนของตัวอักขระได้อย่างมีประสิทธิภาพ เป็นภาษาที่มีข้อความคล้ายภาษาอังกฤษ อย่างไรก็ดีเป็นภาษาที่ต้องใช้หน่วยความจำหลักขนาดใหญ่และยังจำกัดการใช้อยู่กับเครื่องคอมพิวเตอร์ของบริษัท IBM

ภาษาเอพีแอลเป็นภาษาที่ใช้กับระบบอินเทอร์แอคทีฟ มีความสามารถในการคำนวณสูงและสามารถจัดการแถวของตัวอักขระได้อย่างมีประสิทธิภาพสูงด้วยภาษานี้มีโครงสร้างแบบเสรี (free form structure) และมีตัวดำเนินการพิเศษจำนวนมากที่ทำให้การจัดการที่ยุ่งยากบางอย่างไม่ต้องใช้ความพยายามมากเกินไป อย่างไรก็ดีมันต้องการหน่วยความจำหลักขนาดใหญ่ ไม่เป็นภาษามาตรฐานและไม่เป็นภาษาที่มีข้อความคล้ายภาษาอังกฤษด้วย

5. ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์



6. การแบ่งซอฟท์แวร์ตามลักษณะการทำงาน

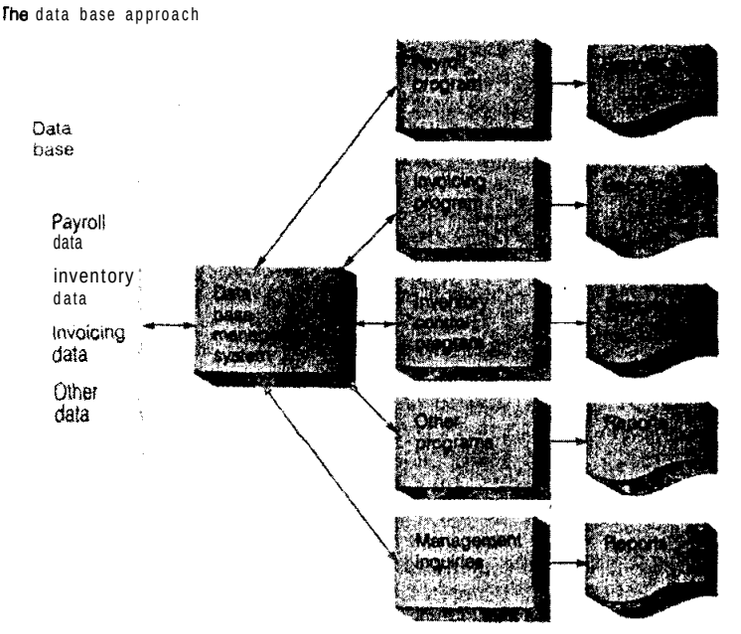
เราอาจแบ่งซอฟท์แวร์ตามลักษณะการทำงานของมันได้ใหญ่ ๆ 2 ชนิด คือ

1. **System software** หรือ System package ซึ่งช่วยให้ระบบคอมพิวเตอร์ดำเนินไปอย่างมีประสิทธิภาพ ซอฟท์แวร์ชนิดนี้ถ้าดีจะทำให้การใช้ Application software ใช้เวลาคอมพิวเตอร์น้อยและความพยายามน้อย ถ้าปราศจากมันแล้ว application software ก็จะทำงานกับระบบคอมพิวเตอร์ไม่ได้ System software ประกอบด้วย ตัวล่ามทั้งหลาย เช่น แอสเซมเบลอร์ (Assembler) คอมไพเลอร์ (Compiler) โปรแกรมควบคุมระบบ (Operating system : OS) Data base management system (DBMS) และ Utility programs เป็นต้น System software มักเขียนด้วยภาษาเครื่อง และเราจะได้รับจากบริษัทผู้ผลิตพร้อมกับเครื่องที่เราเช่าหรือซื้อ ในที่นี้จะกล่าวถึง 3 ตัวหลังพอสังเขป

โปรแกรมควบคุมระบบ (Operating system : OS, Executive program, monitor, supervisor, controller, master control program) เป็นโปรแกรมที่ควบคุมและบริหารการทำงานของระบบคอมพิวเตอร์ทั้งระบบ ตัวอย่างเช่น โปรแกรม DOS หรือ Disk operating system ซึ่งควบคุมการทำงานที่ใช้จานแม่เหล็ก CP/M หรือ Control Program for Microcomputer เป็นโปรแกรมควบคุมระบบที่นิยมมากที่สุดชนิดหนึ่งสำหรับเครื่องไมโครคอมพิวเตอร์ โปรแกรมควบคุมการทำงานของโปรแกรมต่าง ๆ พร้อม ๆ กัน เป็นต้น โปรแกรมประเภทนี้บริษัทผู้ผลิตจะให้มากับระบบคอมพิวเตอร์ที่เราเช่าหรือซื้อ

DBMS เป็นโปรแกรมสำเร็จรูปซึ่งจัดการและดูแลข้อมูลที่จะต้องใช้กับการประมวลผลข้อมูลของการประยุกต์ใช้หลาย ๆ ด้าน นั่นคือโปรแกรมที่จัดการเกี่ยวกับการทำงานกับระบบฐานข้อมูล (Data base) นั่นเอง

ฐานข้อมูล (Data base)

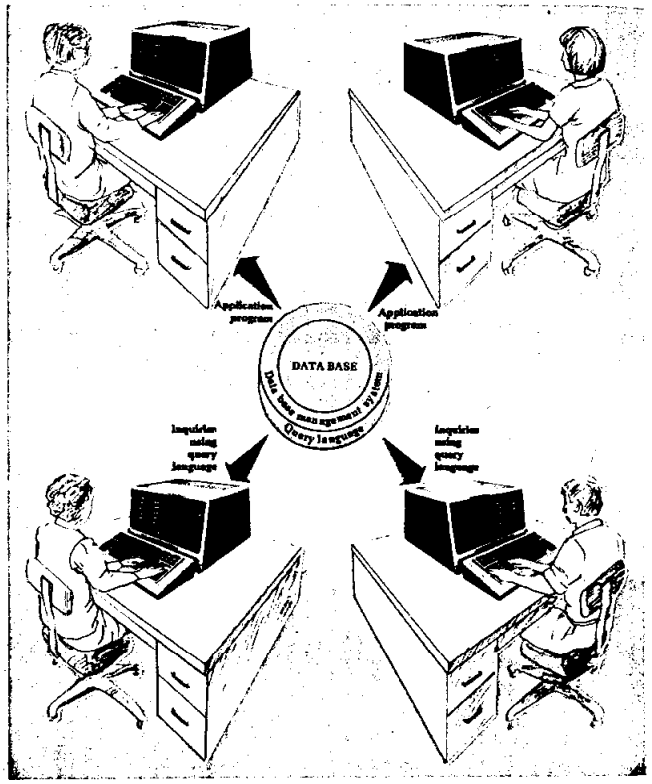


เป็นที่ยอมรับกันว่า ข้อมูลเป็นทรัพยากรที่สำคัญมากขององค์การใด ๆ ถ้าปราศจากข้อมูลและความสามารถในการประมวลผลข้อมูลแล้วองค์การจะอยู่รอดไม่ได้ จะทำให้ไม่สามารถจ่ายค่าจ้างพนักงาน ไม่สามารถส่งใบเก็บเงิน ไม่สามารถสั่งอะไหล่ใหม่เข้ามาสำรองไว้ และไม่สามารถให้ข่าวสารที่ช่วยผู้บริหารในขบวนการตัดสินใจได้

โดยทั่ว ๆ ไปแล้ว ข้อมูลมักจะถูกเก็บไว้ตามบุญตามกรรม เมื่อต้องการรายงานหรือเอกสาร ก็จะเขียนโปรแกรมขึ้นเพื่อการนั้นโดยที่ข้อมูลจะเก็บในเทปแม่เหล็ก บัตรเจาะรูหรือจานแม่เหล็ก โปรแกรมแต่ละโปรแกรมมีแฟ้มข้อมูลของมันเอง แฟ้มข้อมูลเหล่านี้เป็นอิสระต่อกัน เมื่อข้อมูลอย่างเดียวกันจะถูกใช้จากหลาย ๆ โปรแกรม จะต้องทำสำเนาแฟ้มข้อมูลให้มีจำนวนสำเนาเท่ากับโปรแกรมที่จะใช้มัน สิ่งนี้ทำให้เกิดปัญหามาก เมื่อข้อมูลถูกปรับแก้ท่านจะไม่แน่ใจว่าข้อมูลทั้งหมดถูกต้องหรือไม่ เพราะการแก้อาจหลงลืมไม่ได้แก้ไขในทุกสำเนา นอกจากนั้นผู้บริหารจะไม่ได้รับข้อมูลอย่างรวดเร็วและมีประสิทธิภาพจากข้อมูลที่เก็บด้วยระบบคอมพิวเตอร์ วิธีการสร้างและใช้ข้อมูลดังกล่าวข้างต้นเรียกว่า "application approach" ซึ่งมีปัญหาดังกล่าวมาแล้ว ดังนั้น จึงมีการพัฒนาระบบฐานข้อมูลหนึ่งขึ้น ฐานข้อมูลนี้จะรวบรวมข้อมูลสำหรับ

โปรแกรมทุกโปรแกรมและสำหรับตอบคำถามของผู้บริหาร วิธีนี้ไม่ต้องทำสำเนาข้อมูล และการรวบรวมข้อมูลจะดีขึ้นมาก วิธีนี้จะทำให้การป้องกันข้อมูลดีขึ้น ลดจำนวนพนักงาน และพื้นที่ที่ใช้เก็บข้อมูลลง ลดเงินในการดูแลรักษาระบบและลดการลงทุนด้านอุปกรณ์ต่าง ๆ ด้วย ระบบฐานข้อมูลต้องใช้ซอฟท์แวร์พิเศษเพิ่มขึ้นซึ่งจะทำหน้าที่สร้างและดูแลรักษาข้อมูล ซอฟท์แวร์นี้จะทำหน้าที่เหมือนเป็นบัฟเฟอร์ระหว่างโปรแกรมใช้งานต่าง ๆ และฐานข้อมูล ในรูปข้างต้นแสดง "data base approach" ซอฟท์แวร์ที่ใช้คือ DBMS (Data base management system) แฟ้มข้อมูลตามความต้องการของโปรแกรมต่าง ๆ และสำหรับตอบปัญหาของผู้บริหาร ถูกเก็บรวบรวมไว้อย่างมีระเบียบในจานแม่เหล็ก

Utility program คือโปรแกรมที่ทำหน้าที่แต่ละอย่างเป็นประจำวัน เช่น โปรแกรมในการเรียงลำดับข้อมูล (Sort) โปรแกรมในการรวมแฟ้มข้อมูล (merge) หรือโปรแกรมในการทำสำเนาเทป (Copy tape) เป็นต้น โปรแกรมเหล่านี้จะถูกทดลองใช้มาแล้วว่าสามารถทำงานอย่างไม่ผิดพลาด



ระบบการประมวลผลแบบ On-line โดยใช้ฐานข้อมูล (Data Base) ผ่านโปรแกรม Query Language หรือ CICS (Customer Information Control System) โดยมี DBMS (Data Base Management System) ทำหน้าที่ประสานงานระหว่างผู้ใช้กับฐานข้อมูล

2. **Application Software** หรือ Application package เป็นโปรแกรมที่เขียนขึ้นเพื่อแก้ปัญหาหรือสนับสนุนขบวนการตัดสินใจสำหรับบุคคลหรือคณะบุคคลหรือองค์การ โปรแกรมชนิดนี้เราเขียนขึ้นด้วยภาษาในระดับสูง Application package เช่น เงินเดือน ทำบัญชี ควบคุมสินค้าคงคลัง เป็นต้น โปรแกรมเหล่านี้อาจเป็นโปรแกรมสำเร็จรูปที่เขียนขึ้นเพื่อใช้งานบางอย่างดังกล่าวข้างต้นโดยมีบริษัทผู้ผลิต หรือ Software house หรือกลุ่มของผู้ใช้ทำการผลิตออกขายหรือให้เช่า ผู้เขียนโปรแกรมประเภทนี้ต้องมีความชำนาญมาก นอกจากนั้นก็มีโปรแกรมที่ผู้ใช้ (user) เครื่องทำการเขียนขึ้นเองที่เรียกว่า User program โปรแกรมประเภทนี้เขียนขึ้นเพื่อใช้กับงานภายในศูนย์คอมพิวเตอร์โดยเฉพาะ โปรแกรมประเภทนี้อาจมีการให้เปล่ากันได้ ทั้งนี้แล้วแต่ตกลงกัน

ในปัจจุบันโปรแกรมควบคุมระบบนั้นเป็นเรื่องที่ลึกซึ้งมาก มันช่วยทำให้การปฏิบัติงานของระบบคอมพิวเตอร์ดีขึ้นมาก การทำงานของมันเป็นไปอย่างอัตโนมัติจนเรามองข้ามหน้าที่ของมันไป เพราะเรามองไม่เห็นว่ามันกำลังทำอะไรอยู่ เช่น โปรแกรมควบคุมระบบนั้นทำให้ระบบคอมพิวเตอร์วิ่งโปรแกรมหลาย ๆ โปรแกรมอย่างต่อเนื่องกันไปทีละโปรแกรมโดยที่ไม่ต้องอาศัยพนักงานควบคุมเครื่อง (Computer operator) วิ่งโปรแกรมทีละโปรแกรมโดยวิ่งแยกกันทีละโปรแกรม การประมวลผลดังกล่าว คือระบบการประมวลผลแบบกลุ่ม (Batch processing) นั่นเอง นอกจากระบบ Batch แล้วยังมีระบบการทำงานอีกหลายอย่างที่เราควรทราบ

ระบบ Batch เป็นระบบที่ดีมากในการวิ่งโปรแกรม แต่มีงานบางอย่างที่ต้องการการตอบสนองโดยทันทีทันใดจากคอมพิวเตอร์ คำว่า "real-time" แปลว่า การตอบสนองโดยทันทีทันใดจากคอมพิวเตอร์ เช่น การตรวจดูสภาวะสินค้าคงคลังในขณะใดขณะหนึ่ง การค้นหาข้อมูลในแฟ้มข้อมูลอาชญากรรมเพื่อหาผู้ต้องสงสัยซึ่งการกระทำเหล่านี้ต้องการผลหรือคำตอบทันทีทันใดโดยที่จะรอไม่ได้ เมื่อเรามีระบบ real-time นั้น เครื่องมือทุกชนิดจะต้องต่อสายโดยตรงกับเครื่องคอมพิวเตอร์ และรับคำสั่งโดยตรงจากหน่วยประมวลผลกลาง ซึ่งเราใช้คำว่า on-line เพื่ออธิบายสิ่งนี้ ดังนั้นระบบ real time จึงมักจะถูกเรียกว่า On-line real-time system หรือเรียกย่อ ๆ ว่า OLRT นอกจากนั้นถ้ามีผู้ใช้มากกว่า 1 คน (มีหลายเทอร์มินัล มี Card reader หลายเครื่อง เครื่องพิมพ์หลายตัว) ใช้คอมพิวเตอร์ระบบเดียวกันในเวลาเดียวกัน เราเรียกระบบ time sharing

Multiprocessing หมายความถึงตัวโปรเซสเซอร์มากกว่า 1 ตัว นั่นคือใช้ตัวซีพียูมากกว่า 1 ตัวเชื่อมโยงกัน สามารถทำงานเป็นอิสระกันและทำหน้าที่เป็นตัวสำรองซึ่งกันและกัน

มักใช้ในงานสำคัญซึ่งงานจะล้มเหลวไม่ได้เป็นอันขาด นั่นคือถ้าซีพียูตัวใดเสียลง (down) ตัวอื่นก็สามารถทำงานต่อไปได้ ทั้งนี้ เพราะตัวซีพียูทุกตัวจะทำงานชิ้นเดียวกันในเวลาพร้อม ๆ กัน

Multiprogramming หมายความถึงว่าในหน่วยความจำหลักจะเก็บโปรแกรมได้มากกว่าหนึ่งโปรแกรม ทั้งนี้ เพราะหน่วยความจำหลักถูกแบ่งออกเป็นส่วน ๆ เรียกว่า partition แต่ละ partition จะเก็บโปรแกรมที่สมบูรณ์โปรแกรมหนึ่ง ตัวซีพียูทำงานได้เร็วและมีประสิทธิภาพมากกว่าปกติ เพราะโปรแกรมที่จะถูกทำงานไปพร้อม ๆ กันนั้นอยู่ในซีพียูทั้งหมดทุกโปรแกรม ลักษณะการทำงานพร้อม ๆ กันจะเป็นแบบ Overlapped processing ระบบนี้ยึดหลักการที่จะใช้คอมพิวเตอร์ให้เกิดประโยชน์สูงสุดซึ่งต่างจากระบบ time sharing ซึ่งยึดหลักที่จะก่อให้เกิดประโยชน์สูงสุดกับผู้ใช้แต่ละรายในการแก้ปัญหา

Virtual storage เป็นส่วนขยายของ multiprogramming คือแทนที่จะเก็บโปรแกรมที่สมบูรณ์ไว้ในหน่วยความจำหลัก คอมพิวเตอร์จะเก็บบางส่วนของโปรแกรมเท่านั้นในหน่วยความจำหลัก โดยที่ส่วนที่เหลือจะเก็บไว้ในจานแม่เหล็ก ประโยชน์ของ Virtual storage นั้นใช้กับระบบ real-time และระบบ time sharing ซึ่งคำสั่งเพียง 2-3 คำสั่งของผู้ใช้แต่ละคนจะถูกปฏิบัติตามในขณะหนึ่ง ๆ ตัวซีพียูไม่ต้องรอคำสั่งจากผู้ใช้คนอื่น ๆ เพราะบางส่วนของคำสั่งของทุกคนถูกเก็บในหน่วยความจำหลักแล้ว

รูปแสดงแนวคิดของ Virtual storage และ multiprogramming

Multiprogramming

Program-P1
Program-P2
Program-P3

Virtual storage

ส่วนของ Program-P1	ส่วนของ Program-P4
ส่วนของ Program-P2	ส่วนของ Program-P5
ส่วนของ Program-P3	ส่วนของ Program-P6

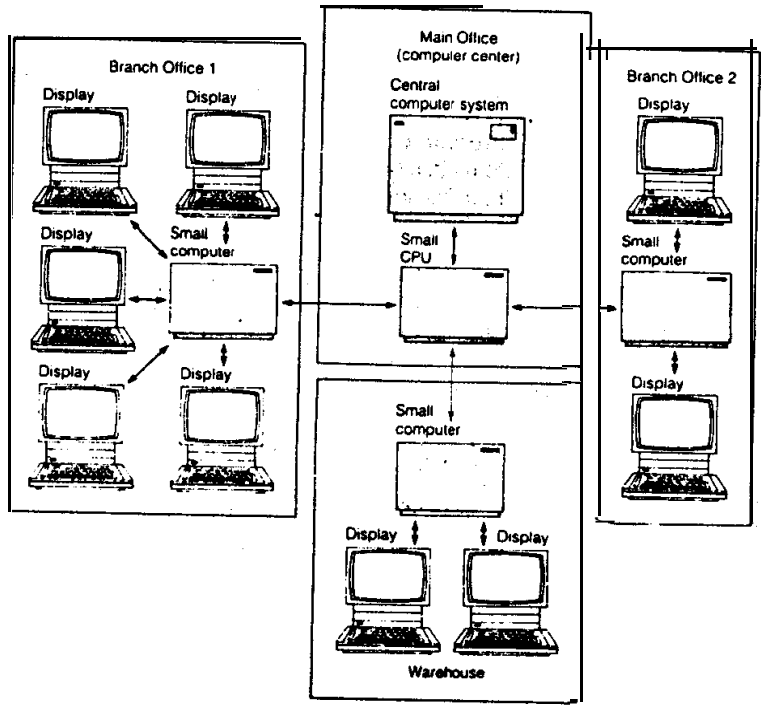
จากรูปแสดงการเก็บโปรแกรมที่สมบูรณ์ 3 โปรแกรมใน 3 partition ของหน่วยความจำหลัก เมื่อใช้ระบบ multiprogramming ส่วนในระบบ Virtual storage จะเก็บส่วนของโปรแกรม 6 โปรแกรมในหน่วยความจำหลัก

Virtual systems เป็นการขยายแนวความคิดของ virtual storage แต่ข้อแตกต่างที่สำคัญคือแนวความคิดนั้นจะถูกนำไปใช้กับทั้งระบบคอมพิวเตอร์ แทนที่จะใช้กับหน่วยความจำเพียงอย่างเดียว ระบบคอมพิวเตอร์ซึ่งใช้ virtual system นั้นยอมให้ผู้ใช้ (users) หลาย ๆ คนเข้าถึงระบบ

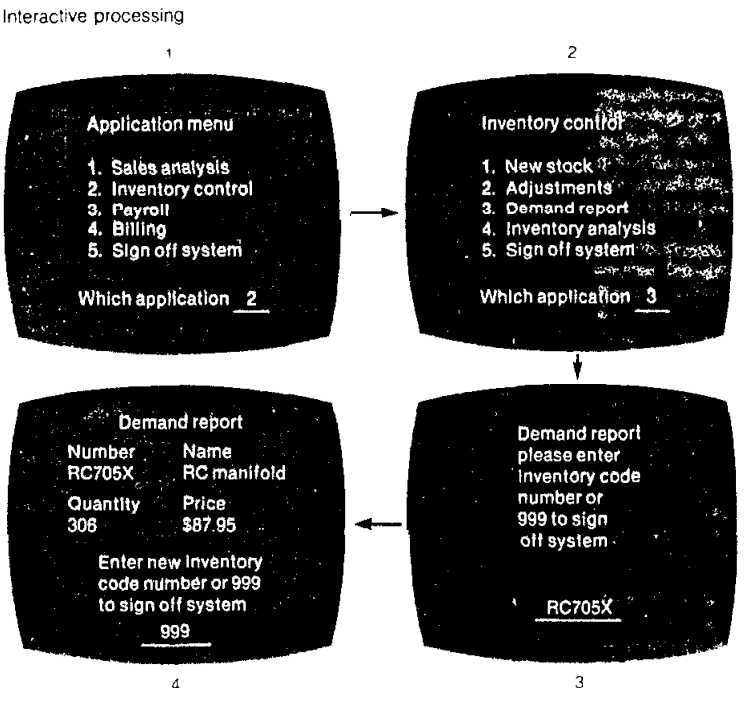
ได้คล้ายกับผู้ใช้ผู้นั้นใช้คอมพิวเตอร์อยู่ผู้เดียวตลอดเวลาที่ใช้คอมพิวเตอร์ คอมพิวเตอร์มีความสามารถที่จะจัดการแบ่งส่วนของระบบอย่างเหมาะสมที่สุดกับผู้ใช้แต่ละคน ขึ้นอยู่กับความต้องการของผู้ใช้ ทรัพยากรทั้งหมดของระบบจะถูกแบ่งอย่างมีประสิทธิภาพที่สุดผู้ใช้ที่ต้องการใช้เครื่องนำข้อมูลเข้า/ออกมากก็จะได้ใช้ในขณะที่คนอื่นจะได้เวลาในการประมวลผลมากขึ้น virtual system นั้นต้องใช้ system software ที่ยุ่งยาก ซอฟท์แวร์ต้องมีความสามารถที่จะรับรู้ความต้องการของผู้ใช้แต่ละคนและหาวิธีที่ดีที่สุดที่จะแบ่งทรัพยากรทั้งหมดที่มีอยู่ซึ่งรวมทั้งฮาร์ดแวร์และซอฟท์แวร์ทั้งหมด จุดประสงค์ก็คือจัดให้ผู้ใช้แต่ละคนได้รับสิ่งที่ตรงกับความต้องการเพื่อทำการประมวลผลข้อมูลที่จะต้องทำ

ระบบอินเทอร์แอคทีฟ (Interactive processing) คือ ระบบผู้ใช้งานสามารถติดต่อโต้ตอบกับเครื่องคอมพิวเตอร์ได้ ระบบจะถามปัญหาผู้ใช้ไปเป็นชุดเพื่อทำงานหนึ่งหรือมากกว่า สิ่งนี้เรียกว่า manu driven system (ดูตัวอย่างจากรูป) เพราะว่าผู้ใช้จะได้รับเมนูหลาย ๆ เมนูให้เลือกคำตอบ ผู้ใช้ไม่ต้องทราบวิธีการเขียนโปรแกรม เพียงแต่คอยตอบปัญหาจากระบบเท่านั้น การประมวลผลวิธีนี้มักจะต้องใช้จอซีอาร์ทีซึ่งคำถามทั้งหมดและคำตอบจะปรากฏบนจอ

A multiprocessing system



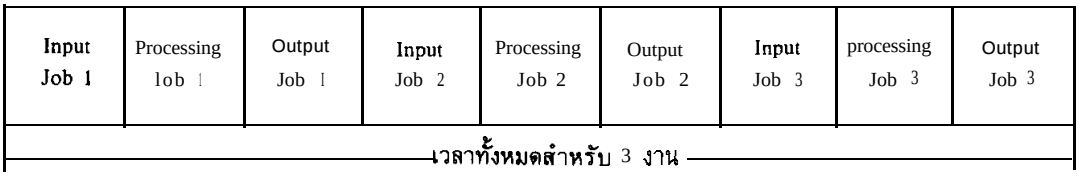
ระบบมัลติโปรเซสซิง



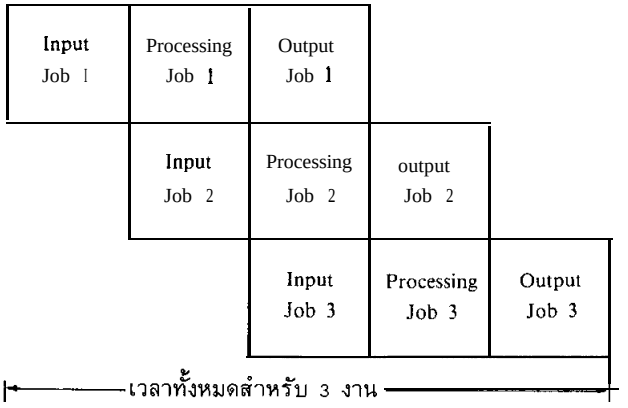
ระบบอินเตอร์แอคทีฟ

Overlapped processing เมื่อหน่วยนำข้อมูลเข้า/ออกกำลังทำงานอย่างรวดเร็วเท่าที่จะเป็นไปได้ และซีพียูไม่ได้ทำงานเป็นเวลานาน ระบบนั้นเราเรียกว่า Input/Output bound ซึ่งคือระบบที่ความเร็วของการทำงานถูกจำกัดด้วยความเร็วของหน่วยนำข้อมูลเข้า/ออก อีกระบบหนึ่งซึ่งตรงกันข้ามเป็นระบบที่ความเร็วของการทำงานถูกจำกัดด้วยความเร็วของซีพียู หรือเป็นระบบที่ใช้เวลาของซีพียูมากกว่าเวลาที่ใช้หน่วยนำข้อมูลเข้า/ออก เราเรียกว่า Process bound ในทั้ง 2 ระบบ การทำงานตามโปรแกรมหนึ่งจะต้องเสร็จสิ้นก่อนที่จะทำงานในโปรแกรมถัดไป แต่ใน overlapped processing นั้น ในเวลาเดียวกันโปรแกรมหนึ่งอาจนำข้อมูลเข้าอีกโปรแกรมหนึ่งอาจใช้ตัวซีพียูทำการประมวลผล และโปรแกรมที่ 3 อาจนำข้อมูลออกแนวความคิดง่าย ๆ นี้ทำให้ลดเวลาของการวิ่งโปรแกรมชุดหนึ่งลงได้ จากรูปจะแสดงการประมวลผลแบบ nonoverlapped และแบบ overlapped ซึ่งจะเห็นว่าเราสามารถประหยัดเวลาลงได้

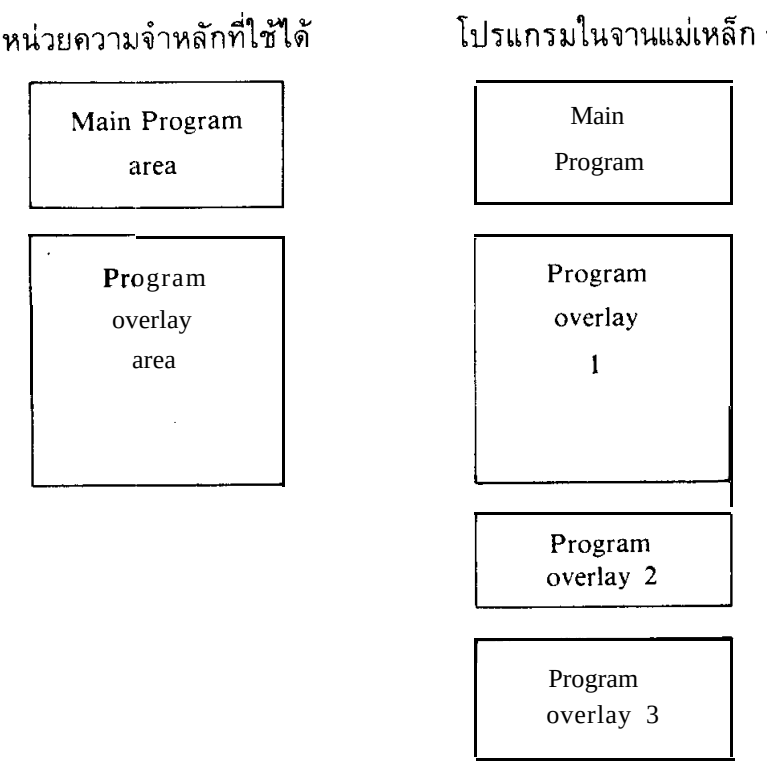
Nonoverlapped processing



Overlapped processing



Program overlays ในบางกรณีโปรแกรมอาจมีขนาดใหญ่และไม่อาจเก็บไว้ในหน่วยความจำหลักได้หมด เพื่อแก้ปัญหานี้ คอมพิวเตอร์บางเครื่องใช้ program overlays program overlays เป็นส่วนของโปรแกรมขนาดใหญ่ เมื่อโปรแกรมเมอร์เขียนโปรแกรม โปรแกรมถูกแบ่งออกเป็นโปรแกรมหลัก (Main program) และหลาย ๆ program overlay เมื่อโปรแกรมถูกปฏิบัติตามโปรแกรมหลักจะยังคงอยู่ในหน่วยความจำหลักตลอดไปจนจบงาน ส่วน program overlay จะถูกอ่านจาก disk เข้าไปเก็บในหน่วยความจำหลักครั้งละ overlay เมื่อ overlay ใหม่เข้าไปอยู่ในหน่วยความจำหลัก overlay เก่าจะถูกส่งกลับไปเก็บในจานแม่เหล็ก ดังนั้น ณ เวลาใด ๆ ในหน่วยความจำหลักจะมีโปรแกรมหลักและ overlay อยู่ 1 overlay เสมอ ดังนั้น โปรแกรมเมอร์จะต้องแน่ใจว่าได้แบ่งโปรแกรมออกเป็นส่วน ๆ โดยที่ส่วนของหน่วยความจำจะต้องเก็บ main program และ overlay ที่ใหญ่ที่สุดได้ รูปต่อไปแสดง Program overlays


หน่วยความจำหลักที่ใช้ได้
โปรแกรมในจานแม่เหล็ก
Main Program area
Program overlay area
Main Program
Program overlay 1
Program overlay 2
Program overlay 3