บทที่ 7
ASSIGNMENTS AND EXPRESSIONS

คำสั่ง assignments ใช้สำหรับกำหนดค่าคงที่ หรือค่าของ expression ให้กับตัวแปร ในคำสั่ง assignment จะมีคีย์เวิร์ด และมีรูปแบบดังนี้

```
variable = expression;
```

เมื่อ variable เป็น scalar element, หรือ อะเรย์ หรือ ชื่อโครงสร้าง หรือตัวแปรเทียม ส่วน expression นั้นจะอยู่ในหัวข้อที่จะกล่าวถึงต่อไป

7.1 Expressions

expression ประกอบด้วย ตัวถูกกระทำ และ ตัวปฏิบัติการ ซึ่งเมื่อมีการคำนวณและที่โปรแกรมทำงาน จะให้ค่าคงที่หนึ่งค่า กฎที่ใช้กำกับการจัดลำดับ ของตัวถูกอ้างถึง (references), ตัวปฏิบัติการ และ วงเล็บ ใช้ใน expression แต่ละชุด

ตัวถูกอ้างถึง (reference) อาจจะเป็นค่าคงที่, ตัวแปร, หรือฟังก์ชัน

ตัวปฏิบัติการ (operator) ใช้ นิยาม การคำนวณ ที่จะกระทำกับตัวกระทำต่าง ๆ

วงเล็บ (parentheses) ใช้บ่งส่วนต่าง ๆ ของ expression

รูปแบบที่ถูกต้องของ ตัวถูกกระทำ, ตัวปฏิบัติการ และวงเล็บ จะได้กล่าวในหัวข้อต่อไปนี้

7.1.1 Prefix expressions

expression ชนิดนี้ ประกอบด้วย ตัวปฏิบัติการ 1 ตัว (unary operator) แล้วตามด้วยตัวถูกกระทำ 1 ตัว ตัวถูกกระทำจะได้รับการประเมินผลก่อน แล้วตัวปฏิบัติการจึงจะ applied ให้กับ ผลลัพธ์นั้น

ตัวอย่าง

A และ -SQRT(B)

7.1.2 Infix expressions

expression ชนิดนี้ ประกอบด้วยตัวถูกกระทำ 2 ตัว คั่นด้วยตัวปฏิบัติการ 1 ตัว ตัวถูกกระทำในที่นี้ อาจจะเป็น expression ก็ได้ และจะได้รับการประเมินผลก่อน จากนั้น ตัวปฏิบัติการจึงจะ applied ให้กับผลลัพธ์นั้น หมายเหตุ ลำดับของการประเมินผล ไม่ได้กำหนดใน PL/I ดังนั้น โปรแกรมคอมไพเลอร์ สามารถเลือกชุดการประเมินผล ที่ให้ผลมากที่สุด

ตัวอย่าง infix expressions

```
A + B       C**2
```

7.1.3 Precedence of operators

ลำดับที่ซึ่งตัวปฏิบัติการจะถูก applied ใน expression ที่ไม่มีวงเล็บกำกับ หรือ subexpression ถูกควบคุมโดย a set precedence ของตัวปฏิบัติการ ลำดับที่กำหนดตายตัวของความสำคัญสูงต่ำ ใน PL/I ได้แสดงไว้แล้ว ข้างล่างนี้ จากตำแหน่งที่มีความสำคัญสูงสุด จนถึง ตำแหน่งที่มีความสำคัญต่ำสุด ส่วนตัวปฏิบัติการที่มีความสำคัญเท่ากัน จะแสดงไว้ในบรรทัดเดียวกัน

ยกกำลัง	**
NOT	~ หรือ ^
prefix operators	+, -
คูณ, หาร	*, /
บวก, ลบ	+, -
concatenation	|| ,!! หรือ \\
relational operators	=, ~=, <, ~<, >, ~>, <=, >=
AND	&
OR	, ! หรือ \

ความสำคัญของตัวปฏิบัติการ จะกระทำในวงเล็บ ที่ตัวปฏิบัติการ มีระดับสูงที่สุด

ก่อน แล้วจึงทำ ตัวปฏิบัติการที่มีระดับต่ำกว่า จนกระทั่ง expression ทั้งหมด ถูกต้องตามวงเล็บที่กำกับ ในกรณีที่ ตัวปฏิบัติการมี ระดับเท่ากัน เกิดในระดับเดียวกัน ตัวปฏิบัติการ prefix และ ยกกำลัง จะถูกประเมินผลจากขวาไปซ้าย ส่วนตัวปฏิบัติการอื่น ๆ ที่เหลือนั้น จะถูกประเมินผลจากซ้ายไปขวา

ตัวอย่าง expression ที่ไม่มีวงเล็บ กำกับ

```
2+Z*X**Y**2/5-q
```

โปรแกรมคอมไพเลอร์ จะปฏิบัติดังนี้

```
(2+((Z*(X**(Y**2)))/5))-Q
```

7.1.4 Relational operators

ตัวปฏิบัติการกลุ่มนี้ ใช้เปรียบเทียบ ค่าที่คำนวณไม่ได้ ว่าเท่ากัน หรือไม่เท่ากัน ค่าที่คำนวณได้ก็อาจนำมาเปรียบเทียบได้เหมือนกัน แต่จะต้องเป็นไปตามกฎพีชคณิตทั่วไป ถ้าตัวถูกกระทำนั้น ๆ มีรูปแบบไม่เหมือนกัน สิ่งแรก มันจะถูกเปลี่ยนรูป (convert) ให้เป็นชนิดเดียวกัน (common type) ซึ่งจะได้กล่าวในรายละเอียดในหัวข้อต่อไป ก่อนที่จะเกิดการเปรียบเทียบกัน ถ้าผลของการเปรียบเทียบนั้นเป็นจริง เครื่องจะให้ bit string ความยาว 1 ค่า '1'B แต่ถ้า ผลของการเปรียบเทียบนั้นเป็นเท็จ เครื่องจะให้ผลลัพธ์เป็น '0'B การเปรียบเทียบ character string ตัวถูกกระทำที่สั้นกว่า เครื่องจะใส่ blanks ให้ทางขวามือ จนกระทั่งมันมีความยาวเท่ากับ ความยาวของตัวถูกกระทำที่ยาวกว่า การเปรียบเทียบกระทำทีละตัวอักขระ (character by character) จากซ้ายไปขวา โดยการใช้ ASCII collating sequence ที่ นิยาม ในภาคผนวก A

การเปรียบเทียบ bit string, bit string ตัวที่สั้นกว่า เครื่องจะใส่ bit 0 ให้ทางขวามือ การเปรียบทียบ ทำทีละบิท (bit by bit) จากซ้ายไปขวา โดยที่ 0 จะมีค่าน้อยกว่า 1

7.1.5 Bit string operators

ตัวปฏิบัติการ bit string มีดังนี้

```
^  ~  |  !  \  &
```

เมื่อ สัญลักษณ์ 2 ตัวแรกแทน ตัวปฏิบัติการ NOT (logical negate) อีก 3 ตัวถัดไปแทน ตัวปฏิบัติการ OR (logical OR) และสัญลักษณ์ตัวสุดท้ายแทน ตัวปฏิบัติการ AND (logical AND)

การปฏิบัติการ bit string ถูกกระทำทีละบิท ตัวปฏิบัติการ NOT จะเปลี่ยนแต่ละบิท ในตัวถูกกระทำ bit string ให้ตรงกันข้าม นั่นคือ บิท 0 แต่ละตัวถูกเปลี่ยนเป็นบิท 1 และบิท 1 แต่ละตัวถูกเปลี่ยนเป็น บิท 0 ตัวปฏิบัติการ OR และตัวปฏิบัติการ AND จะต้องใช้ตัวถูกกระทำ bit string 2 ชุด ถ้าตัวถูกกระทำ 2 ชุดนั้นมีความยาวไม่เท่ากัน ชุดที่สั้นกว่า จะถูกขยายไปทางขวาด้วย บิท 0 จนกระทั่งมันมีความยาวเท่ากับตัวถูกกระทำอีกตัวหนึ่ง ความยาวของ string ผลลัพธ์ จะเท่ากับ ความยาวของตัวถูกกระทำทั้ง 2 ตัวนั้น

ตัวปฏิบัติการ OR และ AND เป็นไปตามกฎของ Boolean algebra ดังนี้

X	Y	X | Y
0	0	0
0	1	1
1	0	1
1	1	1

X	Y	X & Y
0	0	0
0	1	0
1	0	0
1	1	1

(ฟังก์ชัน boolean อื่น ๆ สร้างง่ายมาก โดยใช้ ฟังก์ชัน บิลท์-อิน BOOL)

7.1.6 Exponentiation

การยกกำลัง ถูกคำนวณเป็น ชุดของการคูณ ถ้า exponent เป็น ค่าคงที่ชนิด non-negative นอกเหนือจากนี้ การปฏิบัติการจะถูกประเมินผล โดยการ ใช้ฟังก์ชัน

LOG และ EXP แต่ก็มีกรณีพิเศษของการยกกำลังที่ต้องพิจารณาดังนี้

ถ้า X=0 และ Y>0 แล้ว X**Y=0

ถ้า X=0 และ Y<0 เกิดเงื่อนไข ERROR

ถ้า X~=0 และ Y=0 แล้ว X**Y=1

ถ้า X<0 และ Y ไม่ใช่เป็นเลขจำนวนเต็ม (integer) เกิดเงื่อนไข ERROR

7.2 Arithmetic conversions

การเปลี่ยนรูป ระหว่าง ข้อมูลชนิดต่าง ๆ อาจเกิดขึ้นได้ระหว่างที่เครื่องประเมินผล expression การเปลี่ยนรูปทั้งหมด เกี่ยวข้องกับข้อมูลเดิม (source data), ผลลัพธ์ขั้นกลาง (intermediate result) และ ข้อมูลเป้าหมาย (target data) ผลลัพธ์ขั้นกลาง ถูกคำนวณจากข้อมูลเดิม เป็นไปตามกฎที่จะกล่าวถึงต่อไป และรูปแบบของข้อมูลเป้าหมาย ได้มาจากผลลัพธ์ขั้นกลาง

7.2.1 การเปลี่ยนรูปจากเลขคณิตไปเป็นเลขคณิต

(Arithmetic to arithmetic conversions)

กฎทั่วไปสำหรับ ก้ารเปลี่ยนรูป ตัวถูกกระทำเลขคณิต มีดังนี้

1) ถ้าตัวถูกกระทำตัวหนึ่งเป็น FIXED และตัวถูกกระทำอีกตัวหนึ่งเป็น FLOAT, ชนิดร่วม (commom type) เป็น FLOAT. FIXED BINARY (p) ถูกเปลี่ยนรูปเป็น FLOAT BINARY (p) ขณะที่ FIXED DECIMAL (p,q) ถูกตัดเศษ และเปลี่ยนรูปเป็น FLOAT BINARY (p'), p'=min(ceil(p*3.32),24)

2) ถ้าตัวถูกกระทำตัวหนึ่ง เป็น FIXED BINARY และชนิดของตัวถูกกระทำอีกตัวหนึ่ง เป็น FIXED DECIMAL, ชนิดร่วมจะเป็น FIXED BINARY

FIXED DECIMAL(p,q) จะกลายเป็น FIXED BINARY(p)

หลังจากการเปลี่ยนรูปเป็นชนิดร่วม เกิดขึ้นแล้ว

3) ถ้าตัวถูกกระทำเป็น FLOAT BINARY ผลลัพธ์จะเป็น FLOAT BINARY และ

precision ของผลลัพธ์จะเป็น precision ชุดที่มากกว่าของตัวถูกกระทำสองตัวนั้น

4) ถ้าตัวถูกกระทำเป็น FIXED DECIMAL, ให้ (p,q) เป็น precision ของตัวถูกกระทำตัวแรก และ (r,s) เป็น precision ของตัวถูกกระทำตัวที่สองแล้ว สำหรับการบวก หรือ การลบ จะให้ precision ของผลลัพธ์เป็น

(MIN(15,MAX(p-q,r-s)+MAX(q,s)+1),MAX(q,s))

สำหรับการคูณ precision ของผลลัพธ์ คือ

(MIN(15,p+r+1),q+s)

สำหรับการหาร, precision ของผลลัพธ์ คือ

(15,15-p+q-s)

แต่ควรระวัง เมื่อหารค่า FIXED DECIMAL ฟังก์ชัน DIVIDE อาจนำมาใช้ควบคุม precision ของผลหาร (quotient)

5) ถ้าตัวถูกกระทำ เป็น FIXED BINARY ให้ (p) เป็น precision ของตัวถูกกระทำตัวแรก และ (r) เป็น precision ของตัวถูกกระทำตัวที่สอง แล้ว สำหรับการบวก หรือ การลบ precision ของผลลัพธ์ คือ

(MIN(15,MAX(p,r)+1))

สำหรับการคูณ precision ของผลลัพธ์ คือ

(MIN(15,p+r+1)

บิลท์-อิน ฟังก์ชัน DIVIDE ต้องใช้ให้สอดคล้องกับภาษา PL/I ชุดสมบูรณ์ และต้องกำหนด scale factor ของศูนย์ ให้ผลลัพธ์เป็น integral

การตัดเศษ (truncation) เกิดขึ้นถ้า precision ของผลลัพธ์ ไม่พอเก็บตัวเลข, การตัดเศษเกิดทางขวาของข้อมูล ชนิด FLOAT BINARY ขณะที่เลขหลังจุดทศนิยม (fractional part digits) จะหายไป ในการคำนวณ FIXED DECIMAL

ตัวเลข FIXED BINARY ส่วนหน้าสุดจะหายไป ฟังก์ชัน การเปลี่ยนรูปเลขคณิตจำนวนหนึ่ง ถูกจัดให้ควบคุม การเปลี่ยนรูปที่เกิดขึ้น ระหว่างการประเมินผล expression, รายละเอียดให้ดูในหัวข้อต่อไป

7.2.2 FIXED built-in function

บิลท์-อิน ฟังก์ชัน FIXED ถูกอ้างถึงโดย FIXED(x) หรือ FIXED(x,p) หรือ FIXED(X,p,q) เมื่อ X เป็นตัวแปร หรือ expression ที่จะถูกเปลี่ยนรูป ให้เป็นข้อมูลเลขคณิต ชนิด FIXED, p และ q หมายถึง precision และ scale เป้าหมาย ถ้า X เป็น FIXED DECIMAL ผลลัพธ์จะเป็น FIXED DECIMAL นอกเหนือจากนี้ ผลลัพธ์จะเป็น FIXED BINARY, ถ้า X เป็น FLOAT BINARY ผลลัพธ์จะเป็น FIXED BINARY ถ้าไม่กำหนด p หรือ q แล้ว ผลลัพธ์จะขึ้นกับ precision และ scale ของ X ดังนี้

X FIXED BINARY(r) ให้เป็น FIXED BINARY(r)

X FLOAT BIANRY(r) ให้เป็น FIXED BINARY(MIN(15,r))

X FIXED DECIMAL(r,s) ให้เป็น FIXED DECIMAL(r,s)

7.2.3 FLOAT built-in function

บิลท์-อิน ฟังก์ชัน FLOAT ถูกอ้างถึงโดย FLOAT(X) หรือ FLOAT(X,p) เมื่อ X เป็นตัวแปร หรือ expression ที่ต้องการเปลี่ยนรูป ให้เป็น ข้อมูลเลขคณิตชนิด FLOAT และ p เป็น precision ของเป้าหมาย ถ้าไม่กำหนด p แล้วผลลัพธ์ จะเป็นดังนี้

X FIXED BINARY(r) ให้เป็น FLOAT BINARY(r)

X FLOAT BINARY(r) ให้เป็น FLOAT BINARY(r)

X FIXED DECIMAL (r,s) ให้เป็น

FLOAT BINARY(MIN(CEIL((r-s)*3.32),24))

7.2.4 BINARY built-in function

บิลท์-อิน ฟังก์ชัน BINARY ถูกอ้างถึงโดย BINARY(X) หรือ BIANRY(X,p) เมื่อ X เป็นตัวแปร หรือ expression ที่ต้องการเปลี่ยนรูป ให้เป็น ข้อมูลเลขคณิตชนิด BINARY และ p เป็น precision ของเป้าหมาย

ถ้า X เป็น FIXED BINARY หรือ FIXED DECIMAL ผลลัพธ์จะเป็น FIXED BINARY ถ้า X เป็น FLOAT แล้วผลลัพธ์ จะเป็น FLOAT BINARY ถ้าไม่กำหนด p ผลลัพธ์จะเป็นดังนี้

X FLOAT BINARY (r) ให้เป็น FLOAT BINARY (r)

X FIXED BINARY (r) ให้เป็น FIXED BINARY (r)

X FIXED DECIMAL (r,s) ให้เป็น

FIXED BINARY(MIN(CEIL((r-s)*3.32)+1,15))

7.2.5 DECIMAL built-in function

บิลท์-อิน ฟังก์ชัน DECIMAL ถูกอ้างถึงโดย DECIMAL(x) หรือ DECIMAL(x,p) หรือ DECIMAL(x,p,q) เมื่อ x เป็นตัวแปรหรือ expression ที่ต้องการให้เปลี่ยนรูปให้เป็น ข้อมูล เลขคณิตชนิด FIXED DECIMAL และ p,q เป็น precision และ scale หรือผลลัพธ์เป้าหมาย ถ้าไม่กำหนด p และ q แล้ว ผลลัพธ์จะเป็นดังนี้

X FIXED BINARY(r) ให้เป็น FIXED DECIMAL(CEIL(r/3.32)+1,0)

X FLOAT BINARY(r) ให้เป็น FIXED DECIMAL(MIN(CEIL(r/3.32),15),0)

X FIXED DECIMAL(r,s) ให้เป็น FIXED DECIMAL(r,s)

7.2.6 DIVIDE built-in function

บิลท์-อิน ฟังก์ชันตัวนี้ ใช้เพื่อควบคุม precision ของผลลัพธ์ จากการหาร มีรูปแบบดังนี้

DIVIDE (X,Y,P[,Q])

เมื่อ X และ Y เป็น arithmetic expression, X เป็นตัวตั้ง, Y เป็นตัวหาร, P เป็น expression ชนิด FIXED BINARY หมายถึง precision ที่ต้องการ, Q เป็น expression ชนิด FIXED BINARY หมายถึง scale ที่ต้องการ ถ้าไม่กำหนดค่า Q เครื่องจะถือว่าเป็นศูนย์ ฟังก์ชัน DIVIDE จะต้องเป็นการหารของ FIXED BINARY และในภาษา PL/I ชุดสมบูรณ์ การปฏิบัติการเช่นนี้จะให้ผลลัพธ์เป็น non-zero scale factor

7.3 การเปลี่ยนรูป string (String conversions)

การเปลี่ยนรูปเกิดขึ้นระหว่าง data item ชนิด เลขคณิต และ bit string เมื่อทั้ง 2 ตัวนี้ถูก combined ใน expressions, กฎการเปลี่ยนรูปต่าง ๆ สำหรับตัวถูกกระทำชนิด string จะได้กล่าวถึงในหัวข้อนี้

7.3.1 การเปลี่ยนรูปจากเลขคณิต ให้เป็น bit string

การเปลี่ยนรูปจาก ข้อมูลชนิดเลขคณิต X ให้เป็น bit string เป้าหมายเกิดขึ้นตามกฎต่อไปนี้ ABS(X) ถูกเปลี่ยนรูปเป็น FIXED BINARY(p) ตามกฎการเปลี่ยนรูปเลขคณิต จากนั้นค่า FIXED BINARY ที่เกิดขึ้น จึงถูกเปลี่ยนให้เป็น bit string ความยาว p, ถ้าความยาวเป้าหมายมากกว่า p เครื่องจะเติมบิท 0 ทางขวามือให้กับ ผลลัพธ์ที่เกิดขึ้น แต่ถ้าความยาวเป้าหมาย น้อยกว่า p, เครื่องจะตัดบิทที่เกินทางขวามือสุดของผลลัพธ์ที่เกิดขึ้น

7.3.2 การเปลี่ยนรูปเลขคณิต ให้เป็น character

ข้มูลชนิดเลขคณิตต่าง ๆ จะถูกเปลี่ยนรูป ให้เป็น intermediate character string ในลักษณะต่อไปนี้

FIXED DECIMAL(p,q), q=0; character string ผลลัพธ์ จะมีความยาวเท่ากับ p+3 ตัวอักขระนี้ (characters) จะประกอบด้วย ตัวเลขจาก source ไม่มีเลขศูนย์นำหน้าตัวเลข และ มีเครื่องหมายลบนำหน้า ถ้าค่าของ source เป็นค่าลบ และเครื่องจะเติม blanks ให้ทางซ้ายมือ เพื่อทำให้ character string มีความยาวเท่ากับ p+3

ตัวอย่าง การเปลี่ยนรูป FIXED DECIMAL(3) ที่มีค่าเท่ากับ 330 จะให้ character string ผลลัพธ์เป็น 'bbb330' เมื่อ b หมายถึง ตำแหน่ง blank สำหรับค่าศูนย์ จะให้ผลลัพธ์ เป็นเลขศูนย์ 1 ตัว

FIXED DECIMAL(p,q), q>0 : จะให้ character string ผลลัพธ์ มีความยาวเท่ากับ p+3 เช่นกัน และมีรูปแบบของ string เช่นเดียวกับที่กล่าวข้างต้น ยกเว้น จุดทศนิยม และเลขหลังจุดทศนิยม

ตัวอย่าง การเปลี่ยน FIXED DECIMAL(5,2) ข้อมูลมีค่าเท่ากับ -13.25 จะให้ character string ผลลัพธ์เป็น 'bb-13.25' เลขศูนย์ที่นำหน้าตัวเลขจะไม่มี ยกเว้น เครื่องจะให้เลขศูนย์ 1 ตัว หน้าจุดทศนิยม

FIXED BINARY(b) : source จะถูกเปลี่ยนเป็น FIXED DECIMAL(p) เมื่อ p = CEIL(b/3.32)+1 จากนั้น FIXED DECIMAL(p) ผลลัพธ์ จะถูกเปลี่ยนให้เป็น character string ความยาว p+3 มีรูปแบบเช่นที่กล่าวข้างต้น

ตัวอย่าง การเปลี่ยนรูป FIXED BINARY(15) ที่มีค่าเท่ากับ -32 จะให้ผลลัพธ์เป็น 'bbbbbb-32'

FLOAT BINARY(b) : ส่วนที่เป็นเลขหลังจุดทศนิยมจะถูกเปลี่ยนรูป ให้เป็น FIXED DECIMAL(p) เมื่อ p = CEIL(b/3.32) จะให้ character string ผลลัพธ์ ความยาวเท่ากับ p+6 และมีรูปแบบของ สัญลักษณ์ทางวิทยาศาสตร์มาตรฐาน (standard scientific notation) ดังนี้ ถ้าค่าของ source เป็นลบ character ตัวแรก จะเป็นเครื่องหมายลบ แต่ถ้าค่าของ source ไม่เป็นลบ ตำแหน่งแรกนั้น จะเป็น blank 1 ตัว ตำแหน่งถัดไป จะประกอบด้วย ตัวเลขหน้าสุดของค่านั้น ตามด้วย จุดทศนิยม และเลขหลังจุดทศนิยมส่วนที่เหลือ p-1 ตัว ตามด้วย exponent indicator E เครื่องหมาย exponent และค่าของ exponent เป็นเลขอีก 2 ตัว

ตัวอย่าง การเปลี่ยนรูป ของ FLOAT BINARY (24) ที่มีค่าเท่ากับ 250.1E1 จะให้ character string ผลลัพธ์เป็น 'bbbb2.501E+03'

ถ้าความยาวเป้าหมาย มากกว่าความยาวของ ผลลัพธ์ที่ได้ เครื่องจะเติม blank ให้ทางขวามือของ string นั้น แต่ในทางตรงกันข้ามถ้า ความยาวเป้าหมาย น้อยกว่าความยาวของผลลัพธ์ที่ได้ เครื่องจะตัด ทางขวามือของผลลัพธ์เพื่อให้ความยาวเท่ากัน

7.3.3 การเปลี่ยนรูป bit string ให้เป็น arithmetic

bit string ที่มีความยาวเท่ากับ n เมื่อ 0 < n <= 15 เมื่อเปลี่ยนรูปให้เป็นข้อมูลชนิดเลขคณิต เริ่มแรก จะถูกเปลี่ยนให้เป็น FIXED BINARY(15) ที่มีความหมายเหมือนกัน จากนั้นจึงเปลี่ยนให้เป็น ค่าเป้าหมาย (target value) ตามกฎที่ได้กล่าวมาแล้วในหัวข้อก่อนหน้านี้

ตัวอย่าง bit string '1011'B เมื่อเปลี่ยนให้เป็น FIXED BINARY(15) จะให้ค่าเท่ากับ 11

7.3.4 การเปลี่ยนรูป bit string ให้เป็น character string

bit string ที่มีความยาวเท่ากับ n จะถูกเปลี่ยนรูป ให้เป็น character string ความยาว n เมื่อ บิท 0 จะถูกเปลี่ยนให้เป็น ตัวอักขระ 0 และบิท 1 จะถูกเปลี่ยนให้เป็น ตัวอักขระ 1 ถ้าความยาวเป้าหมาย มากกว่า source เครื่องจะใส่ blank ทางขวามือ ให้กับเป้าหมาย แต่ถ้า ความยาวเป้าหมาย น้อยกว่าความยาวของ source ตัวอักขระทางขวามือสุด ส่วนที่เกินจะถูกตัดทิ้ง

7.3.5 การเปลี่ยนรูป character ให้เป็น arithmetic

ในกรณีของการเปลี่ยนรูป character ให้เป็น arithmetic, source ที่เป็น character string ต้องประกอบด้วย ค่าคงที่ตัวเลขที่ถูกต้อง (valid arithmetic constant value) ถ้า X เป็น character string การเปลี่ยนรูปที่ให้กับ X มีผลมาจาก การเปลี่ยนรูปเลขคณิต ของฟังก์ชัน บิลท์-อิน ดังนี้

FIXED(X) หรือ DECIMAL(X) ให้ผลลัพธ์เป็นค่า FIXED DECIMAL ถ้าไม่กำหนดค่า p เครื่องจะถือว่าเป็น 15, BINARY(X) ให้ค่า FIXED BINARY ถ้าไม่กำหนดค่า p เครื่องจะ set ให้เป็น 15

หมายเหตุ ผลลัพธ์ จะให้เฉพาะส่วนที่เป็นเลขจำนวนเต็มของ X เท่านั้น

FLOAT(X) ให้ผลลัพธ์เป็นค่า FLOAT BINARY ถ้าไม่กำหนดค่า p เครื่องจะถือว่าเท่ากับ 24 ถ้า X ไม่มีค่า (null) หรือมีแต่ blanks ทั้งหมด, ค่าที่เปลี่ยนรูปแล้วจะเป็นศูนย์

เงื่อนไข ERROR จะเกิดขึ้น ถ้า character string ไม่อยู่ในรูปแบบที่ถูกต้องทางเลขคณิต หรือถ้า ขนาดของข้อมูลเป้าหมาย ไม่พอที่จะเก็บค่าที่เปลี่ยนรูปแล้ว

ตัวอย่าง การเปลี่ยนรูปจาก character ให้เป็นข้อมูลชนิดเลขคณิต

character	เป้าหมาย	ผลลัพธ์
'00987'	FIXED BINARY(15)	987
'9.87'	FIXED DECIMAL(6,2)	0009.87
'-9.87E2'	FLOAT BINARY(24)	-9.87E2
'-9.87E2'	FIXED DECIMAL(9,2)	0000987.00
'-9.87E2'	FIXED DECIMAL(5,0)	00987
'-987.372'	FIXED DECIMAL(4,2)	ERROR
'2I3'	FIXED DECIMAL(15)	ERROR

7.3.6 การเปลี่ยนรูปจาก character ให้เป็น bit string

ในกรณีของการเปลี่ยนรูปจาก character ให้เป็น bit string, character string ที่เป็น source ต้องมีเฉพาะตัวอักขระ (character) 0 และ 1 เท่านั้น ตัวอักขระ 0 แต่ละตัว จะถูกเปลี่ยนให้เป็น บิท 0 และตัวอักขระ 1 แต่ละตัวจะถูกเปลี่ยนให้เป็น บิท 1 ถ้าความยาวเป้าหมาย มากกว่า ความยาวของ source เครื่องจะเติม 0 ให้ทางขวามือ แต่ถ้าความยาวเป้าหมาย น้อยกว่า ความยาวของ source เครื่องจะตัดส่วนที่เกินทางขวามือทิ้ง ถ้า source เป็น null string หรือเป็น banks ทั้งหมด แล้ว ผลลัพธ์จะเป็น bit string ที่มีแต่ 0

7.4 ตัวแปรเทียม (Pseudo-variables)

ชื่อ บิลท์-อิน 2 ตัวคือ SUBSTR และUNSPEC ที่ได้กล่าวมาแล้วใน โปรแกรม PL/I-80 ทั้งหมด สามารถนำมาใช้เป็น ตัวถูกกระทำ source (source oerand) ใน expressions หรือ ตัวถูกกระทำเป้าหมาย (target operands) ทางซ้ายมือ ของคำสั่ง assignments ได้ ชื่อ 2 ตัวนี้เป็น ตัวแปรเทียม เนื่องจากมันทำหน้าที่ได้เช่นเดียวกับ ตัวแปรธรรมดาในโปรแกรม

7.4.1 Character SUBSTR

ตัวปฏิบัติการ (operator) ชนิด character substring ใช้ในการเข้าถึง (access) ตัวอักขระแต่ละตัวที่อยู่ภายใน string โดยมีรูปแบบ 2 ชนิดคือ

SUBSTR(char-variable,i)

และ SUBSTR(char-variable,i,j)

เมื่อ char-variable เป็น การอ้างถึง ตัวแปร ชนิด CHAR หรือ CHAR VARYING มี subscripted หรือ ตัวแปรที่ไม่มี subscript

i และ j เป็น FIXED BINARY expressions

เมื่อ SUBSTR ปรากฏใน expression ใด, รูปแบบแรก คือการแบ่งเอา (extracts) substring ที่ตำแหน่ง i จนกระทั่งจบส่วนที่เหลือของ string ชุดนั้น โดยที่ตำแหน่งของตัวอักขระตัวแรก ใน string จะมีหมายเลขเป็น 1 ส่วนรูปแบบที่สอง ทำหน้าที่เช่นเดียวกับรูปแบบแรก แต่ ความยาวของ substring ที่แบ่งมานั้น เท่ากับ j

หมายเหตุ ผลลัพธ์จะไม่รู้ค่า (undefined) ถ้า i หรือ i+j มากกว่าความยาวของ string เมื่อความยาวของ ตัวแปรชนิด CHAR นั้น ต้อง declare ด้วยขนาดคงที่ (fixed size) และเป็นความยาวปัจจุบัน ของ ตัวแปรชนิด CHAR VARYING

เมื่อ SUBSTR ปรากฏทางซ้ายมือของคำสั่ง assignment, ต้องปรากฏเพียงตัวเดียว นั่นคือ การปฏิบัติการ SUBSTR ไม่สามารถจัดกลุ่ม(embedded)ใน string expression เมื่อมันทำหน้าที่เป็น เป้าหมายของ string assignment และการปฏิบัติการ SUBSTR ที่ปรากฏ ในโปรแกรม จะมีลักษณะดังนี้

```
SUBSTR(char-variable,i) = char-exp;
SUBSTR(char-variable,i,j) = char-exp;
```

รูปแบบแรก คือการกำหนด character expression ให้เป็น substring ใน char-variable เริ่มตั้งแต่ ตำแหน่ง i และขยาย จนถึงความยาวของ char-exp หรือจนจบความยาวของ char-variable อย่างใดอย่างหนึ่งแล้วแต่ว่าอันไหนจะเกิดก่อน ส่วนรูปแบบที่สอง ให้ผลอย่างเดียวกัน ยกเว้น ความกว้างของ field ซึ่งรับตัวอักขระถูกจำกัด ให้มีความยาวเท่ากับ j

หมายเหตุ ค่าของ i และ i+j ต้องอยู่ในช่วง ความยาวปัจจุบัน หรือความยาวคงที่ของ string มิฉะนั้นแล้ว การปฏิบัติการนี้ จะให้ผลลัพธ์เป็น ค่าที่ไม่ทราบ (undefined)

นอกจากนี้ char-variable ตัวเดียวกันนี้ อาจปรากฏ ทั้งทางซ้ายมือและขวามือ ของคำสั่ง assignment โดยไม่ต้องมี partial substring overwrite ระหว่าง assignment ซึ่งลักษณะของการใช้เช่นนี้ อาจปรากฏได้ใน PL/I

ตัวอย่าง

```
SUBSTR(C(I),J,K+2)=SUBSTR(D,J)||SUBSTR(E,J+5,3);
```

7.4.2 Bit SUBSTR

การปฏิบัติการชนิด bit substring ใน PL/I-80 คล้ายกับ การปฏิบัติการชนิด character SUBSTR ที่กล่าวมาแล้วข้างต้น แต่มีข้อจำกัดบางอย่าง สิ่งแรก bit strings ของ PL/I-80 ถูกจำกัดให้มี precision อยู่ในช่วง 1 ถึง 16, สมนัย (corresponding) กับ ค่า หนึ่ง ไบท์ หรือ สอง ไบท์ เพื่อการจัดทำ account

สำหรับค่า intermediate precision ระหว่างการดำเนินการ ความยาวของ การปฏิบัติการ ชนิด bit substring ต้องคงที่ โดยมีรูปแบบ ดังนี้

```
SUBSTR(bit-variable,k)
และ  SUBSTR(bit-variable,i,k)
```

เมื่อ bit-variable เป็น การอ้างถึงตัวแปร ชนิด BIT ที่มี subscript หรือไม่มี subscript ก็ได้, k เป็น ค่าคงที่ ในช่วง 1 ถึง 16 และ i เป็น FIXED BINARY expression ผลของการปฏิบัติการ SUBSTR จะเหมือนกับ การปฏิบัติการ character ที่กล่าวมาแล้วข้างต้น ยกเว้น bit string ที่มีความยาวเท่ากับ k จะถูกเพิ่ม เมื่อ SUBSTR ปรากฏใน expression และถูกตัดทอน เมื่อ SUBSTR ปรากฏทางซ้ายมือ ให้เป็นเป้าหมายของ การปฏิบัติการเก็บ bit string ตัวอย่างของ bit SUBSTR อยู่ในหัวข้อถัดไป

7.4.3 UNSPEC

ตัวแปรเทียม UNSPEC เป็นการเข้าถึง ตัวแปรชนิด หนึ่งไบท์ หรือ สองไบท์ (double bytes) ราวกับว่า มันเป็น ข้อมูลชนิด bit string ที่เป็น 8 บิท และ 16 บิท รูปแบบ การอ้างถึง มีดังนี้

```
UNSPEC (variable)
```

เมื่อ variable เป็น ตัวแปรที่มี subscript หรือไม่มี subscript ก็ได้ อ้างถึง ข้อมูล (data item) ซึ่งเก็บในตำแหน่งของหน่วยความจำหนึ่งไบท์ หรือสองไบท์

หมายเหตุ ผลลัพธ์ชั่วคราว จะนำไปใช้เป็น argument ไม่ได้

เมื่อ UNSPEC อยู่ใน expression, ผลลัพธ์ คือ ค่า bit string ของ argument

เมื่อ UNSPEC ปรากฏทางซ้ายมือของ คำสั่ง assignment ค่าที่ assign จะถูกเปลี่ยนรูป ให้เป็น bit string และเก็บ โดยตรงในค่า หนึ่งไบท์ หรือ สองไบท์

ตัวแปรเทียม UNSPEC จะถูกนำมาใช้บ่อย, เป็นกลไกของ "escape" เมื่อคุณลักษณะปกติ ของภาษานั้น ไม่ปรากฏให้เข้าถึง underlying facilities โปรดระวัง อย่างไรก็ตาม novic โปรแกรมเมอร์ มักจะตกหลุมพลาง ของ การใช้ UNSPEC บ่อย ๆ แทนที่จะเป็น facility ของภาษาระดับสูงที่นำมาใช้ได้มากกว่า ความจริงแล้ว เมื่อใดก็ตาม ที่มี ความจำเป็นต้องใช้ UNSPEC โปรแกรมเมอร์ ควร ตรวจปัญหา ในวิธีทั่ว ๆ ไป มากกว่า เพื่อดูว่า สามารถหลีกเลี่ยงการใช้ได้หรือไม่

ตัวอย่าง

```
DCL
   (P,Q) POINTER,
   A  BIT (16) BASED(P),
   B BIT (8) BASED(Q),
   T  FIXED;
      :
   UNSPEC (P) = 'FF80'b4
   UNSPEC (Q) = 'FFF0'b4;
      :
   SUBSTR(B,4,2) = SUBSTR(A,I,2)
      :
```

หมายถึง ตำแหน่งในหน่วยความจำสองแห่ง กำลังถูก เข้าถึง การปฏิบัติการ UNSPEC ใช้ โหลด absolute address ให้กับ ตัวแปร พอยท์เตอร์ 2 ตัว ตัวแปรชนิด based ทั้งสองจะแบ่ง (overlay) ตำแหน่ง ในหน่วยความจำสองแห่งนี้ เพื่อที่ว่า จะได้ เข้าถึง ปริมาณ 16 บิท และ 8 บิท จากนั้น ตัวแปรเทียม bit SUBSTR จะถูก applied เพื่อย้าย substring จากตำแหน่งหนึ่ง ไปยังตำแหน่งอื่น