Shift

SSI: 부호를 사용한 워드 우 쉬프트

설 명

Accu.1 우측 워드(비트 0 – 15)를 우측으로 비트 단위로 쉬프트하는데 “부호를 사용한 워드 우 쉬프트” 명령을 사용한다. 쉬프트 동안에, 비게되는 자리들은 비트 15(INT 수의 부호 비트)의 신호상태로 채워진다. Accu.1의 비트 16 – 31은 변경되지 않는다.

비트들이 쉬프트 되는 비트 자리들의 수를 지정하는데 다음과 같은 옵션들이 있다.:

- 명령의 파라메타로서 양의 정수 값. 지정되는 숫자는 0 – 15 사이의 값이어야 한다.
- Accu.2 우측 바이트의 값. 이 바이트 값은 양의 정수로서 해석된다.

명령은 RLO와 상관없이 실행된다. 명령은 상태 비트들 CC0와 OV를 “0”으로 리세트하고 상태 비트 CC1을 최종 쉬프트되어 빠져나간 비트의 신호상태로 세트한다.

쉬프트되는 자리수가 15보다 크면, Accu.1 우측 워드의 모든 비트들은 비트 15의 신호상태로 채워진다.

쉬프트되는 자리수가 0이면, 명령은 여전히 실행된다.. 상태 비트 CC1은 “0”으로 리세트 된다.

문 장

쉬프트되는 자리 수가 “부호를 사용한 워드 우 쉬프트” 명령에 대한 파라메타로서 지정되면, 다음과 같은 문장을 사용한다.:

```
SSI       <쉬프트 되는 자리 수>
```

쉬프트되는 자리 수가 Accu.2 우측 바이트를 사용하여 지정되면, “부호를 사용한 워드 우 쉬프트” 명령에 대해 다음과 같은 문장을 사용한다.:

```
SSI
```

파라메타

아래의 표에 명령의 파라메타들이 나타나 있다.

파라메타	데이터 형	설 명
<쉬프트되는 자리수>	양수 SINT, INT,DINT, USINT, UINT, UDINT	쉬프트되는 비트 자리 수.

예

아래의 예에서는 명령이 동작하는 원리가 소개된다.:

STL		설 명
L	“Tag_Value_1”	// Load value of the operand in accumulator 1.
SSI	6	// Shift bits 0 to 15 of accumulator 1 six positions to the right. // Fill the now empty bit places with the state of bit 15.
T	“Tag_Result_1”	// Transfer contents of accumulator 1 to the operand.
L	3	// Load number of places shifted into accumulator 1
L	“Tag_Value_2”	// Move number of places shifted to the right byte of accumulator 2. // Load value of the operand in accumulator 1.
SSI		// Shift bits 0 to 15 of accumulator 1 three positions to the right. // Fill the now empty bit places with the state of bit 15.
T	“Tag_Result_2”	// Transfer contents of accumulator 1 to the operand.

다음 표에서는 특정 오퍼런드 값들을 사용한 명령의 동작 원리가 소개된다.:

오퍼런드	값							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Tag_Result_1	0101	1111	0110	0100	1111	1110	0111	0100
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Result_2	0101	1111	0110	0100	0000	1011	1010	0101

SSD: 부호를 사용한 더블워드 우 쉬프트

설 명

Accu.1의 전체 내용을 우측으로 비트 단위로 쉬프트하는데 “부호를 사용한 더블워드 우 쉬프트” 명령을 사용한다. 쉬프트 동안에, 비게되는 자리들은 비트 31(DINT 수의 부호 비트)의 신호상태로 채워진다.

비트들이 쉬프트 되는 비트 자리들의 수를 지정하는데 다음과 같은 옵션들이 있다.:

- 명령의 파라메타로서 양의 정수 값. 지정되는 숫자는 0 – 31 사이의 값이어야 한다.
- Accu.2 우측 바이트의 값. 이 바이트 값은 양의 정수로서 해석된다.

명령은 RLO와 상관없이 실행된다. 명령은 상태 비트들 CC0와 OV를 “0”으로 리세트하고 상태 비트 CC1을 최종 쉬프트되어 빠져나간 비트의 신호상태로 세트한다.

쉬프트되는 자리수가 31보다 크면, Accu.1의 모든 비트들은 비트 31의 신호상태로 채워진다.

쉬프트되는 자리수가 0이면, 명령은 여전히 실행된다.. 상태 비트 CC1은 “0”으로 리세트 된다.

문 장

쉬프트되는 자리 수가 “부호를 사용한 더블워드 우 쉬프트” 명령에 대한 파라메타로서 지정되면, 다음과 같은 문장을 사용한다.:

SSD <쉬프트 되는 자리 수>

쉬프트되는 자리 수가 Accu.2 우측 바이트를 사용하여 지정되면, “부호를 사용한 더블워드 우 쉬프트” 명령에 대해 다음과 같은 문장을 사용한다.:

SSD

파라메타

아래의 표에 명령의 파라메타들이 나타나 있다.

파라메타	데이터 형	설 명
<쉬프트되는 자리수>	양수 SINT, INT,DINT, USINT, UINT, UDINT	쉬프트되는 비트 자리 수.

예

아래의 예에서는 명령이 동작하는 원리가 소개된다. :

STL		설 명
L	“Tag_Value_1”	// Load value of the operand in accumulator 1.
SSD	7	// Shift bits 0 to 31 of accumulator 1 seven positions to the right. // Fill the now empty bit places with the state of bit 31.
T	“Tag_Result_1”	// Transfer contents of accumulator 1 to the operand.
L	4	// Load number of places shifted into accumulator 1
L	“Tag_Value_2”	// Move number of places shifted to the right byte of accumulator 2. // Load value of the operand in accumulator 1.
SSD		// Shift bits 0 to 31 of accumulator 1 four positions to the right. // Fill the now empty bit places with the state of bit 31.
T	“Tag_Result_2”	// Transfer contents of accumulator 1 to the operand.

다음 표에서는 특정 오퍼런드 값들을 사용한 명령의 동작 원리가 소개된다.:

오퍼런드	값							
Tag_Value_1	1000	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	1111	1111	0001	1110	1100	1000	1011	1010
Tag_Value_2	0010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	0000	0010	1000	1010	0010	1001	1011	1100

SLW: 워드 좌 쉬프트

설 명

Accu.1 우측 워드(비트 0 – 15)를 좌측으로 비트 단위로 쉬프트하는데 “워드 좌 쉬프트” 명령을 사용한다. 쉬프트 동안에, 비게되는 자리들은 “0”으로 채워진다. Accu.1의 비트 16 – 31은 변경되지 않는다.

비트들이 쉬프트 되는 비트 자리들의 수를 지정하는데 다음과 같은 옵션들이 있다.:

- 명령의 파라메타로서 양의 정수 값. 지정되는 숫자는 0 – 15 사이의 값이어야 한다.
- Accu.2 우측 바이트의 값. 이 바이트 값은 양의 정수로서 해석된다.

명령은 RLO와 상관없이 실행된다. 명령은 상태 비트들 CC0와 OV를 “0”으로 리세트하고 상태 비트 CC1을 최종 쉬프트되어 빠져나간 비트의 신호상태로 세트한다.

쉬프트되는 자리수가 15보다 크면, Accu.1 우측 워드의 모든 비트들은 “0”으로 채워진다.

쉬프트되는 자리수가 0이면, 명령은 여전히 실행된다.. 상태 비트 CC1은 “0”으로 리세트 된다.

문 장

쉬프트되는 자리 수가 파라메타로서 지정되면, “워드 좌 쉬프트” 명령에 대해 다음과 같은 문장을 사용한다.:

SLW <쉬프트 되는 자리 수>

쉬프트되는 자리 수가 Accu.2 우측 바이트를 사용하여 지정되면, “워드 좌 쉬프트” 명령에 대해 다음과 같은 문장을 사용한다.:

SLW

파라메타

아래의 표에 명령의 파라메타들이 나타나 있다.:

파라메타	데이터 형	설 명
<쉬프트되는 자리수>	양수 SINT, INT,DINT, USINT, UINT, UDINT	쉬프트되는 비트 자리 수.

예

아래의 예에서는 명령이 동작하는 원리가 소개된다.:

```
STL                 설 명
L     “Tag_Value_1”  // Load value of the operand in accumulator 1.
SLW   5              // Shift bits 0 to 15 of accumulator 1 five positions
                     to the left.
                     // Fill the now empty bit places with zeros.
T     “Tag_Result_1” // Transfer contents of accumulator 1 to the operand.
L     4              // Load number of places shifted into accumulator 1
L     “Tag_Value_2”  // Move number of places shifted to the right byte of
                     accumulator 2.
                     // Load value of the operand in accumulator 1.
SLW                  // Shift bits 0 to 15 of accumulator 1 four positions
                     to the left.
                     // Fill the now empty bit places with zeros.
T     “Tag_Result_2” // Transfer contents of accumulator 1 to the operand.
```

다음 표에서는 특정 오퍼런드 값들을 사용한 명령의 동작 원리가 소개된다.:

오퍼런드	값							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	0101	1111	0110	0100	1010	0111	0110	0000
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Result_2	0101	1111	0110	0100	1101	0010	1011	0000

SRW: 워드 우 쉬프트

설 명

Accu.1 우측 워드(비트 0 – 15)를 우측으로 비트 단위로 쉬프트하는데 “워드 우 쉬프트” 명령을 사용한다. 쉬프트 동안에, 비게되는 자리들은 “0”으로 채워진다. Accu.1의 비트 16 – 31은 변경되지 않는다.

비트들이 쉬프트 되는 비트 자리들의 수를 지정하는데 다음과 같은 옵션들이 있다.:

- 명령의 파라메타로서 양의 정수 값. 지정되는 숫자는 0 – 15 사이의 값이어야 한다.
- Accu.2 우측 바이트의 값. 이 바이트 값은 양의 정수로서 해석된다.

명령은 RLO와 상관없이 실행된다. 명령은 상태 비트들 CC0와 OV를 “0”으로 리세트하고 상태 비트 CC1을 최종 쉬프트되어 빠져나간 비트의 신호상태로 세트한다.

쉬프트되는 자리수가 15보다 크면, Accu.1 우측 워드의 모든 비트들은 “0”으로 채워진다.

쉬프트되는 자리수가 0이면, 명령은 여전히 실행된다.. 상태 비트 CC1은 “0”으로 리세트 된다.

문 장

쉬프트되는 자리 수가 파라메타로서 지정되면, “워드 우 쉬프트” 명령에 대해 다음과 같은 문장을 사용한다.:

SRW <쉬프트 되는 자리 수>

쉬프트되는 자리 수가 Accu.2 우측 바이트를 사용하여 지정되면, “워드 우 쉬프트” 명령에 대해 다음과 같은 문장을 사용한다.:

SRW

파라메타

아래의 표에 명령의 파라메타들이 나타나 있다.

파라메타	데이터 형	설 명
<쉬프트되는 자리수>	양수 SINT, INT,DINT, USINT, UINT, UDINT	쉬프트되는 비트 자리 수.

예

아래의 예에서는 명령이 동작하는 원리가 소개된다. :

STL		설 명
L	“Tag_Value_1”	// Load value of the operand in accumulator 1.
SRW	6	// Shift bits 0 to 15 of accumulator 1 six positions to the right. // Fill the now empty bit places with zeros.
T	“Tag_Result_1”	// Transfer contents of accumulator 1 to the operand.
L	4	// Load number of places shifted into accumulator 1
L	“Tag_Value_2”	// Move number of places shifted to the right byte of accumulator 2. // Load value of the operand in accumulator 1.
SRW		// Shift bits 0 to 15 of accumulator 1 four positions to the right. // Fill the now empty bit places with zeros.
T	“Tag_Result_2”	// Transfer contents of accumulator 1 to the operand.

다음 표에서는 특정 오퍼런드 값들을 사용한 명령의 동작 원리가 소개된다.:

오퍼런드	값							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	0101	1111	0110	0100	0000	0001	0111	0100
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Result_2	0101	1111	0110	0100	0000	0101	1101	0010

SLD: 더블워드 좌 쉬프트

설 명

Accu.1의 전체 내용을 좌측으로 비트 단위로 쉬프트하는데 “더블워드 좌 쉬프트” 명령을 사용한다. 쉬프트 동안에, 비게되는 자리들은 “0”으로 채워진다.

비트들이 쉬프트 되는 비트 자리들의 수를 지정하는데 다음과 같은 옵션들이 있다.:

- 명령의 파라메타로서 양의 정수 값. 지정되는 숫자는 0 – 31 사이의 값이어야 한다.
- Accu.2 우측 바이트의 값. 이 바이트 값은 양의 정수로서 해석된다.

명령은 RLO와 상관없이 실행된다. 명령은 상태 비트들 CC0와 OV를 “0”으로 리세트하고 상태 비트 CC1을 최종 쉬프트되어 빠져나간 비트의 신호상태로 세트한다.

쉬프트되는 자리수가 31보다 크면, Accu.1 우측 워드의 모든 비트들은 “0”으로 채워진다.

쉬프트되는 자리수가 0이면, 명령은 여전히 실행된다.. 상태 비트 CC1은 “0”으로 리세트 된다.

문 장

쉬프트되는 자리 수가 파라메타로서 지정되면, “더블워드 좌 쉬프트” 명령에 대해 다음과 같은 문장을 사용한다.:

SLD <쉬프트 되는 자리 수>

쉬프트되는 자리 수가 Accu.2 우측 바이트를 사용하여 지정되면, “더블워드 좌 쉬프트” 명령에 대해 다음과 같은 문장을 사용한다.:

SLD

파라메타

아래의 표에 명령의 파라메타들이 나타나 있다.

파라메타	데이터 형	설 명
<쉬프트되는 자리수>	양수 SINT, INT,DINT, USINT, UINT, UDINT	쉬프트되는 비트 자리 수.

예

아래의 예에서는 명령이 동작하는 원리가 소개된다. :

STL		설 명
L	“Tag_Value_1”	// Load value of the operand in accumulator 1.
SLD	5	// Shift bits 0 to 31 of accumulator 1 five positions to the left. // Fill the now empty bit places with zeros.
T	“Tag_Result_1”	// Transfer contents of accumulator 1 to the operand.
L	4	// Load number of places shifted into accumulator 1

STL		설 명
L	“Tag_Value_2”	// Move number of places shifted to the right byte of accumulator 2. // Load value of the operand in accumulator 1.
SLD		// Shift bits 0 to 31 of accumulator 1 four positions to the left. // Fill the now empty bit places with zeros.
T	“Tag_Result_2”	// Transfer contents of accumulator 1 to the operand.

다음 표에서는 특정 오퍼런드 값들을 사용한 명령의 동작 원리가 소개된다.:

오퍼런드	값							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	1110	1100	1000	1011	1010	0111	0110	0000
Tag_Value_2	1010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	1000	1010	0010	1001	1011	1100	1101	0000

SRD: 더블워드 우 쉬프트

설 명

Accu.1의 전체 내용을 우측으로 비트 단위로 쉬프트하는데 “더블워드 우 쉬프트” 명령을 사용한다. 쉬프트 동안에, 비게되는 자리들은 “0”으로 채워진다.

비트들이 쉬프트 되는 비트 자리들의 수를 지정하는데 다음과 같은 옵션들이 있다.:

- 명령의 파라메타로서 양의 정수 값. 지정되는 숫자는 0 – 31 사이의 값이어야 한다.
- Accu.2 우측 바이트의 값. 이 바이트 값은 양의 정수로서 해석된다.

명령은 RLO와 상관없이 실행된다. 명령은 상태 비트들 CC0와 OV를 “0”으로 리세트하고 상태 비트 CC1을 최종 쉬프트되어 빠져나간 비트의 신호상태로 세트한다.

쉬프트되는 자리수가 15보다 크면, Accu.1 우측 워드의 모든 비트들은 “0”으로 채워진다.

쉬프트되는 자리수가 0이면, 명령은 여전히 실행된다. 상태 비트 CC1은 “0”으로 리세트 된다.

문 장

쉬프트되는 자리 수가 파라메타로서 지정되면, “더블워드 우 쉬프트” 명령에 대해 다음과 같은 문장을 사용한다.:

SRD <쉬프트 되는 자리 수>

쉬프트되는 자리 수가 Accu.2 우측 바이트를 사용하여 지정되면, “더블워드 우 쉬프트” 명령에 대해 다음과 같은 문장을 사용한다.:

SRD

파라메타

아래의 표에 명령의 파라메타들이 나타나 있다.

파라메타	데이터 형	설 명
<쉬프트되는 자리수>	양수 SINT, INT,DINT, USINT, UINT, UDINT	쉬프트되는 비트 자리 수.

예

아래의 예에서는 명령이 동작하는 원리가 소개된다.:

STL		설 명
L	“Tag_Value_1”	// Load value of the operand in accumulator 1.
SRD	7	// Shift bits 0 to 31 of accumulator 1 seven positions to the right. // Fill the now empty bit places with zeros.
T	“Tag_Result_1”	// Transfer contents of accumulator 1 to the operand.
L	4	// Load number of places shifted into accumulator 1
L	“Tag_Value_2”	// Move number of places shifted to the right byte of accumulator 2. // Load value of the operand in accumulator 1.
SRD		// Shift bits 0 to 31 of accumulator 1 four positions to the right. // Fill the now empty bit places with zeros.
T	“Tag_Result_2”	// Transfer contents of accumulator 1 to the operand.

다음 표에서는 특정 오퍼런드 값들을 사용한 명령의 동작 원리가 소개된다.:

오퍼런드	값							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	0000	0000	1011	1110	1100	1000	1011	1010
Tag_Value_2	1010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	0000	1010	1000	1010	0010	1001	1011	1100