ADD

```
AR      R1,R2       [RR]

 ┌────────┬────┬────┐
 │  '1A'  │ R1 │ R2 │
 └────────┴────┴────┘
 0         8   12  15
```

```
AGR      R1,R2       [RRE]

 ┌────────────────┬────────┬────┬────┐
 │     'B908'     │////////│ R1 │ R2 │
 └────────────────┴────────┴────┴────┘
 0                16       24   28  31
```

```
AGFR      R1,R2       [RRE]

 ┌────────────────┬────────┬────┬────┐
 │     'B918'     │////////│ R1 │ R2 │
 └────────────────┴────────┴────┴────┘
 0                16       24   28  31
```

```
A       R1,D2(X2,B2)      [RX]

 ┌────────┬────┬────┬────┬────────────┐
 │  '5A'  │ R1 │ X2 │ B2 │     D2     │
 └────────┴────┴────┴────┴────────────┘
 0         8   12   16   20          31
```

```
AY      R1,D2(X2,B2)      [RXY]

 ┌────────┬────┬────┬────┬────┬────────┬────────┐
 │  'E3'  │ R1 │ X2 │ B2 │DL2 │  DH2   │  '5A'  │
 └────────┴────┴────┴────┴────┴────────┴────────┘
 0         8   12   16   20   32       40      47
```

```
AG       R1,D2(X2,B2)      [RXY]

 ┌────────┬────┬────┬────┬────┬────────┬────────┐
 │  'E3'  │ R1 │ X2 │ B2 │DL2 │  DH2   │  '08'  │
 └────────┴────┴────┴────┴────┴────────┴────────┘
 0         8   12   16   20   32       40      47
```

```
AGF       R1,D2(X2,B2)      [RXY]

 ┌────────┬────┬────┬────┬────┬────────┬────────┐
 │  'E3'  │ R1 │ X2 │ B2 │DL2 │  DH2   │  '18'  │
 └────────┴────┴────┴────┴────┴────────┴────────┘
 0         8   12   16   20   32       40      47
```

The second operand is added to the first operand, and the sum is placed at the first-operand location. For ADD (AR, A, and AY), the operands and the sum are treated as 32-bit signed binary integers. For ADD (AGR, AG), they are treated as 64-bit signed binary integers. For ADD (AGFR, AGF), the second operand is treated as a 32-bit signed binary integer, and the first operand and the sum are treated as 64-bit signed binary integers.

When there is an overflow, the result is obtained by allowing any carry into the sign-bit position and ignoring any carry out of the sign-bit position, and condition code 3 is set. If the fixed-point-overflow mask is one, a program interruption for fixed-point overflow occurs.

The displacement for A is treated as a 12-bit unsigned binary integer. The displacement for AY, AG, and AGF is treated as a 20-bit signed binary integer.

Resulting Condition Code:

0	Result zero; no overflow
1	Result less than zero; no overflow
2	Result greater than zero; no overflow
3	Overflow

Program Exceptions:

- Access (fetch, operand 2 of A, AY, AG, and AGF only)
- Fixed-point overflow
- Operation (AY, if the long-displacement facility is not installed)

ADD HALFWORD

```
AH      R1,D2(X2,B2)      [RX]

 ┌────────┬────┬────┬────┬────────────┐
 │  '4A'  │ R1 │ X2 │ B2 │     D2     │
 └────────┴────┴────┴────┴────────────┘
 0         8   12   16   20          31
```

```
AHY      R1,D2(X2,B2)      [RXY]

 ┌────────┬────┬────┬────┬────┬────────┬────────┐
 │  'E3'  │ R1 │ X2 │ B2 │DL2 │  DH2   │  '7A'  │
 └────────┴────┴────┴────┴────┴────────┴────────┘
 0         8   12   16   20   32       40      47
```

ADD HALFWORD IMMEDIATE

```
AHI      R1,I2      [RI]

 ┌────────┬────┬────┬────────────┐
 │  'A7'  │ R1 │'A' │     I2     │
 └────────┴────┴────┴────────────┘
 0         8   12   16          31
```

```
AGHI      R1,I2      [RI]

 ┌────────┬────┬────┬────────────┐
 │  'A7'  │ R1 │'B' │     I2     │
 └────────┴────┴────┴────────────┘
 0         8   12   16          31
```

The second operand is added to the first operand, and the sum is placed at the first-operand location. The second operand is two bytes in length and is treated as a 16-bit signed binary integer. For ADD HALFWORD (AH, AHY) and ADD HALFWORD IMMEDIATE (AHI), the first operand and the sum are treated as 32-bit signed binary integers. For ADD HALFWORD IMMEDIATE (AGHI), they are treated as 64-bit signed binary integers.

When there is an overflow, the result is obtained by allowing any carry into the sign-bit position and ignoring any carry out of the sign-bit position, and condition code 3 is set. If the fixed-point-overflow mask is one, a program interruption for fixed-point overflow occurs.

The displacement for AH is treated as a 12-bit unsigned binary integer. The displacement for AHY is treated as a 20-bit signed binary integer.

Resulting Condition Code:

0	Result zero; no overflow
1	Result less than zero; no overflow
2	Result greater than zero; no overflow
3	Overflow

Program Exceptions:

- Access (fetch, operand 2 of AH, AHY)
- Fixed-point overflow
- Operation (AHY, if the long-displacement facility is not installed)

Programming Note: An example of the use of the ADD HALFWORD instruction is given in Appendix A, “Number Representation and Instruction-Use Examples.”

ADD LOGICAL

```
ALR     R₁,R₂     [RR]

┌────────┬────┬────┐
│  '1E'  │ R₁ │ R₂ │
└────────┴────┴────┘
0        8    12   15

ALGR     R₁,R₂     [RRE]

┌────────────────┬────────┬────┬────┐
│     'B90A'     │////////│ R₁ │ R₂ │
└────────────────┴────────┴────┴────┘
0                16       24   28  31

ALGFR     R₁,R₂     [RRE]

┌────────────────┬────────┬────┬────┐
│     'B91A'     │////////│ R₁ │ R₂ │
└────────────────┴────────┴────┴────┘
0                16       24   28  31

AL     R₁,D₂(X₂,B₂)     [RX]

┌────────┬────┬────┬────┬────────────┐
│  '5E'  │ R₁ │ X₂ │ B₂ │     D₂     │
└────────┴────┴────┴────┴────────────┘
0        8    12   16   20           31

ALY     R₁,D₂(X₂,B₂)     [RXY]

┌────────┬────┬────┬────┬────┬────────┬────────┐
│  'E3'  │ R₁ │ X₂ │ B₂ │DL₂ │  DH₂   │  '5E'  │
└────────┴────┴────┴────┴────┴────────┴────────┘
0        8    12   16   20   32       40       47

ALG     R₁,D₂(X₂,B₂)     [RXY]

┌────────┬────┬────┬────┬────┬────────┬────────┐
│  'E3'  │ R₁ │ X₂ │ B₂ │DL₂ │  DH₂   │  '0A'  │
└────────┴────┴────┴────┴────┴────────┴────────┘
0        8    12   16   20   32       40       47

ALGF     R₁,D₂(X₂,B₂)     [RXY]

┌────────┬────┬────┬────┬────┬────────┬────────┐
│  'E3'  │ R₁ │ X₂ │ B₂ │DL₂ │  DH₂   │  '1A'  │
└────────┴────┴────┴────┴────┴────────┴────────┘
0        8    12   16   20   32       40       47
```

The second operand is added to the first operand, and the sum is placed at the first-operand location. For ADD LOGICAL (ALR, AL, ALY), the operands and the sum are treated as 32-bit unsigned binary integers. For ADD LOGICAL (ALGR, ALG), they are treated as 64-bit unsigned binary integers. For ADD LOGICAL (ALGFR, ALGF) the second operand is treated as a 32-bit unsigned binary integer, and the first operand and the sum are treated as 64-bit unsigned binary integers.

The displacement for AL is treated as a 12-bit unsigned binary integer. The displacement for ALY, ALG, and ALGF is treated as a 20-bit signed binary integer.

Resulting Condition Code:

0	Result zero; no carry
1	Result not zero; no carry
2	Result zero; carry
3	Result not zero; carry

Program Exceptions:

- Access (fetch, operand 2 of AL, ALY, ALG, and ALGF only)
- Operation (ALY, if the long-displacement facility is not installed)

ADD LOGICAL WITH CARRY

```
ALCR    R₁,R₂     [RRE]

 ┌────────────────┬────────┬────┬────┐
 │     'B998'     │////////│ R₁ │ R₂ │
 └────────────────┴────────┴────┴────┘
 0                16       24   28  31

ALCGR    R₁,R₂     [RRE]

 ┌────────────────┬────────┬────┬────┐
 │     'B988'     │////////│ R₁ │ R₂ │
 └────────────────┴────────┴────┴────┘
 0                16       24   28  31

ALC     R₁,D₂(X₂,B₂)     [RXY]

 ┌────────┬────┬────┬────┬────┬────────┬────────┐
 │  'E3'  │ R₁ │ X₂ │ B₂ │DL₂ │  DH₂   │  '98'  │
 └────────┴────┴────┴────┴────┴────────┴────────┘
 0        8    12   16   20   32       40      47

ALCG      R₁,D₂(X₂,B₂)     [RXY]

 ┌────────┬────┬────┬────┬────┬────────┬────────┐
 │  'E3'  │ R₁ │ X₂ │ B₂ │DL₂ │  DH₂   │  '88'  │
 └────────┴────┴────┴────┴────┴────────┴────────┘
 0        8    12   16   20   32       40      47
```

The second operand and the carry are added to the first operand, and the sum is placed at the first-operand location. For ADD LOGICAL WITH CARRY (ALCR, ALC), the operands, the carry, and the sum are treated as 32-bit unsigned binary integers. For ADD LOGICAL WITH CARRY (ALCGR, ALCG), they are treated as 64-bit unsigned binary integers.

Resulting Condition Code:

0	Result zero; no carry
1	Result not zero; no carry
2	Result zero; carry
3	Result not zero; carry

Program Exceptions:

- Access (fetch, operand 2 of ALC and ALCG only)

Programming Notes:

1. A carry is represented by a one value of bit 18 of the current PSW. Bit 18 is the leftmost bit of the two-bit condition code in the PSW. Bit 18 is set to one by an execution of an ADD LOGICAL or ADD LOGICAL WITH CARRY instruction that produces a carry out of bit position 0 of the result.
2. ADD and ADD LOGICAL may provide better performance than ADD LOGICAL WITH CARRY, depending on the model.

AND

```
NR     R₁,R₂     [RR]

 ┌────────┬────┬────┐
 │  '14'  │ R₁ │ R₂ │
 └────────┴────┴────┘
 0        8    12  15

NGR     R₁,R₂     [RRE]

 ┌────────────────┬────────┬────┬────┐
 │     'B980'     │////////│ R₁ │ R₂ │
 └────────────────┴────────┴────┴────┘
 0                16       24   28  31

N      R₁,D₂(X₂,B₂)     [RX]

 ┌────────┬────┬────┬────┬────────────┐
 │  '54'  │ R₁ │ X₂ │ B₂ │     D₂     │
 └────────┴────┴────┴────┴────────────┘
 0        8    12   16   20          31

NY     R₁,D₂(X₂,B₂)     [RXY]

 ┌────────┬────┬────┬────┬────┬────────┬────────┐
 │  'E3'  │ R₁ │ X₂ │ B₂ │DL₂ │  DH₂   │  '54'  │
 └────────┴────┴────┴────┴────┴────────┴────────┘
 0        8    12   16   20   32       40      47

NG      R₁,D₂(X₂,B₂)     [RXY]

 ┌────────┬────┬────┬────┬────┬────────┬────────┐
 │  'E3'  │ R₁ │ X₂ │ B₂ │DL₂ │  DH₂   │  '80'  │
 └────────┴────┴────┴────┴────┴────────┴────────┘
 0        8    12   16   20   32       40      47

NI     D₁(B₁),I₂        [SI]

 ┌────────┬────────┬────┬────────────┐
 │  '94'  │   I₂   │ B₁ │     D₁     │
 └────────┴────────┴────┴────────────┘
 0        8        16   20          31
```

- Specification

```
1.-6.  Exceptions with the same priority as the priority of program-
       interruption conditions for the general case.

7.A    Access exceptions for second instruction halfword.

7.B    Operation exception.

8.     Specification exception due to invalid function code
       or invalid register number.

9.     Specification exception due to invalid operand length.

10.    Condition code 0 due to second-operand length originally zero.

11.    Access exceptions for an access to the parameter block, first,
       or second operand.

12.    Condition code 0 due to normal completion (second-operand
       length originally nonzero, but stepped to zero).

13.    Condition code 3 due to partial completion (second-operand
       length still nonzero).
```

Figure 7-28. Priority of Execution: KM and KMC

Programming Notes:

1. When condition code 3 is set, the general registers containing the operand addresses and length, and, for CIPHER MESSAGE WITH CHAINING, the chaining value in the parameter block, are usually updated such that the program can simply branch back to the instruction to continue the operation.

   For unusual situations, the CPU protects against endless reoccurrence of the no-progress case and also protects against setting condition code 3 when the portion of the first and second operands to be reprocessed overlap in storage. Thus, the program can safely branch back to the instruction whenever condition code 3 is set with no exposure to an endless loop and no exposure to incorrectly retrying the instruction.

2. If the length of the second operand is nonzero initially and condition code 0 is set, the registers are updated in the same manner as for condition code 3. For CIPHER MESSAGE WITH CHAINING, the chaining value in this case is such that additional operands can be processed as if they were part of the same chain.

3. To save storage, the first and second operands may overlap exactly or the starting point of the first operand may be to the left of the starting point of the second operand. In either case, the overlap is not destructive.

COMPARE

```
CR     R₁,R₂     [RR]

┌────────┬────┬────┐
│  '19'  │ R₁ │ R₂ │
└────────┴────┴────┘
0        8   12   15

CGR     R₁,R₂     [RRE]

┌────────────────┬────────┬────┬────┐
│     'B920'     │////////│ R₁ │ R₂ │
└────────────────┴────────┴────┴────┘
0                16       24   28  31

CGFR     R₁,R₂     [RRE]

┌────────────────┬────────┬────┬────┐
│     'B930'     │////////│ R₁ │ R₂ │
└────────────────┴────────┴────┴────┘
0                16       24   28  31
```

```
C      R1,D2(X2,B2)     [RX]

 ┌────────┬────┬────┬────┬────────────┐
 │  '59'  │ R1 │ X2 │ B2 │     D2     │
 └────────┴────┴────┴────┴────────────┘
 0         8   12   16   20           31
```

```
CY     R1,D2(X2,B2)     [RXY]

 ┌────────┬────┬────┬────┬────┬────────┬────────┐
 │  'E3'  │ R1 │ X2 │ B2 │DL2 │  DH2   │  '59'  │
 └────────┴────┴────┴────┴────┴────────┴────────┘
 0         8   12   16   20   32       40      47
```

```
CG      R1,D2(X2,B2)     [RXY]

 ┌────────┬────┬────┬────┬────┬────────┬────────┐
 │  'E3'  │ R1 │ X2 │ B2 │DL2 │  DH2   │  '20'  │
 └────────┴────┴────┴────┴────┴────────┴────────┘
 0         8   12   16   20   32       40      47
```

```
CGF      R1,D2(X2,B2)     [RXY]

 ┌────────┬────┬────┬────┬────┬────────┬────────┐
 │  'E3'  │ R1 │ X2 │ B2 │DL2 │  DH2   │  '30'  │
 └────────┴────┴────┴────┴────┴────────┴────────┘
 0         8   12   16   20   32       40      47
```

The first operand is compared with the second operand, and the result is indicated in the condition code. For COMPARE (CR, C, CY), the operands are treated as 32-bit signed binary integers. For COMPARE (CGR, CG), they are treated as 64-bit signed binary integers For COMPARE (CGFR, CGF), the second operand is treated as a 32-bit signed binary integer, and the first operand is treated as a 64-bit signed binary integer.

The displacement for C is treated as a 12-bit unsigned binary integer. The displacement for CY, CG, and CGF is treated as a 20-bit signed binary integer.

Resulting Condition Code:

0	Operands equal
1	First operand low
2	First operand high
3	--

Program Exceptions:

- Access (fetch, operand 2 of C, CY, CG, and CGF only)
- Operation (CY, if the long-displacement facility is not installed)

COMPARE AND FORM CODEWORD

```
CFC    D2(B2)          [S]

 ┌────────────────┬────┬────────────┐
 │     'B21A'     │ B2 │     D2     │
 └────────────────┴────┴────────────┘
 0                16   20           31
```

General register 2 contains an index, which is used along with contents of general registers 1 and 3 to designate the starting addresses of two fields in storage, called the first and third operands. The first and third operands are logically compared, and a codeword is formed for use in sort/merge algorithms.

The second-operand address is not used to address data. Bits 49-62 of the second-operand address, with one rightmost and one leftmost zero appended, are used as a 16-bit index limit. Bit 63 of the second-operand address is the operand-control bit. When bit 63 is zero, the codeword is formed from the high operand; when bit 63 is one, the codeword is formed from the low operand. The remainder of the second-operand address is ignored.

General registers 1 and 3 contain the base addresses of the first and third operands. Bits 48-63 of general register 2 are used as an index for addressing both the first and third operands. General registers 1, 2, and 3 must all initially contain even values; otherwise, a specification exception is recognized.

In the access-register mode, access register 1 specifies the address space containing the first and third operands.

The size of the units by which the first and third operands are compared, the size of the resulting codeword, and the participation of bits 0-31 of general registers 1, 2, and 3 in the operation depend on the addressing mode. In the 24-bit or 31-bit addressing mode, the comparison unit is two bytes, the codeword is four bytes, and bits 0-31 are ignored and remain unchanged. In the 64-bit addressing mode, the comparison unit is six bytes, the codeword is eight bytes, and bits 0-31 are used in and may be changed by the operation.

Operation in the 24-Bit or 31-Bit Addressing Mode

INSERT PROGRAM MASK

```
IPM    R1                [RRE]

 ┌────────────────┬────────┬────┬────┐
 │     'B222'     │////////│ R1 │////│
 └────────────────┴────────┴────┴────┘
 0               16       24   28  31
```

The condition code and program mask from the current PSW are inserted into bit positions 34 and 35 and 36-39, respectively, of general register R_1. Bits 32 and 33 of the register are set to zeros; bits 0-31 and 40-63 are left unchanged.

Condition Code: The code remains unchanged.

Program Exceptions: None.

LOAD

```
LR     R1,R2     [RR]

 ┌────────┬────┬────┐
 │  '18'  │ R1 │ R2 │
 └────────┴────┴────┘
 0        8   12  15
```

```
LGR     R1,R2     [RRE]

 ┌────────────────┬────────┬────┬────┐
 │     'B904'     │////////│ R1 │ R2 │
 └────────────────┴────────┴────┴────┘
 0               16       24   28  31
```

```
LGFR     R1,R2     [RRE]

 ┌────────────────┬────────┬────┬────┐
 │     'B914'     │////////│ R1 │ R2 │
 └────────────────┴────────┴────┴────┘
 0               16       24   28  31
```

```
L      R1,D2(X2,B2)     [RX]

 ┌────────┬────┬────┬────┬────────────┐
 │  '58'  │ R1 │ X2 │ B2 │     D2     │
 └────────┴────┴────┴────┴────────────┘
 0        8   12   16   20           31
```

```
LY     R1,D2(X2,B2)     [RXY]

 ┌────────┬────┬────┬────┬────┬────────┬────────┐
 │  'E3'  │ R1 │ X2 │ B2 │DL2 │  DH2   │  '58'  │
 └────────┴────┴────┴────┴────┴────────┴────────┘
 0        8   12   16   20   32       40       47
```

```
LG     R1,D2(X2,B2)     [RXY]

 ┌────────┬────┬────┬────┬────┬────────┬────────┐
 │  'E3'  │ R1 │ X2 │ B2 │DL2 │  DH2   │  '04'  │
 └────────┴────┴────┴────┴────┴────────┴────────┘
 0        8   12   16   20   32       40       47
```

```
LGF     R1,D2(X2,B2)     [RXY]

 ┌────────┬────┬────┬────┬────┬────────┬────────┐
 │  'E3'  │ R1 │ X2 │ B2 │DL2 │  DH2   │  '14'  │
 └────────┴────┴────┴────┴────┴────────┴────────┘
 0        8   12   16   20   32       40       47
```

The second operand is placed unchanged at the first-operand location, except that, for LOAD (LGFR, LGF), it is sign extended.

For LOAD (LR, L, LY), the operands are 32 bits, and, for LOAD (LGR, LG), the operands are 64 bits. For LOAD (LGFR, LGF), the second operand is treated as a 32-bit signed binary integer, and the first operand is treated as a 64-bit signed binary integer.

The displacement for L is treated as a 12-bit unsigned binary integer. The displacement for LY, LG, and LGF is treated as a 20-bit signed binary integer.

Condition Code: The code remains unchanged.

Program Exceptions:

- Access (fetch, operand 2 of L, LY, LG, and LGF only)
- Operation (LY, if the long-displacement facility is not installed)

Programming Note: An example of the use of the LOAD instruction is given in Appendix A, "Number Representation and Instruction-Use Examples."

LOAD ACCESS MULTIPLE

```
LAM     R1,R3,D2(B2)     [RS]

 ┌────────┬────┬────┬────┬────────────┐
 │  '9A'  │ R1 │ R3 │ B2 │     D2     │
 └────────┴────┴────┴────┴────────────┘
 0        8   12   16   20           31
```

```
LAMY     R1,R3,D2(B2)     [RSY]

 ┌────────┬────┬────┬────┬────┬────────┬────────┐
 │  'EB'  │ R1 │ R3 │ B2 │DL2 │  DH2   │  '9A'  │
 └────────┴────┴────┴────┴────┴────────┴────────┘
 0        8   12   16   20   32       40       47
```

The set of access registers starting with access register R_1 and ending with access register R_3 is loaded from the locations designated by the second-operand address.

The storage area from which the contents of the access registers are obtained starts at the location designated by the second-operand address and continues through as many storage words as the number of access registers specified. The access registers are loaded in ascending order of their register numbers, starting with access register R_1 and continuing up to and including access register R_3, with access register 0 following access register 15.

The displacement for LAM is treated as a 12-bit unsigned binary integer. The displacement for LAMY is treated as a 20-bit signed binary integer.

The second operand must be designated on a word boundary; otherwise, a specification exception is recognized.

Condition Code: The code remains unchanged.

Program Exceptions:

- Access (fetch, operand 2)
- Operation (LAMY, if the long-displacement facility is not installed)
- Specification

LOAD ADDRESS

```
LA     R1,D2(X2,B2)     [RX]

┌─────────┬────┬────┬────┬────────────┐
│  '41'   │ R1 │ X2 │ B2 │     D2     │
└─────────┴────┴────┴────┴────────────┘
0         8    12   16   20           31
```

```
LAY    R1,D2(X2,B2)     [RXY]

┌─────────┬────┬────┬────┬────/────┬─────────┬─────────┐
│  'E3'   │ R1 │ X2 │ B2 │DL2 │ DH2     │  '71'   │
└─────────┴────┴────┴────┴────/────┴─────────┴─────────┘
0         8    12   16   20   32        40        47
```

The address specified by the X_2, B_2, and D_2 fields is placed in general register R_1. The address computation follows the rules for address arithmetic.

In the 24-bit addressing mode, the address is placed in bit positions 40-63, bits 32-39 are set to zeros, and bits 0-31 remain unchanged. In the 31-bit addressing mode, the address is placed in bit positions 33-63, bit 32 is set to zero, and bits 0-31 remain unchanged. In the 64-bit addressing mode, the address is placed in bit positions 0-63.

The displacement for LA is treated as a 12-bit unsigned binary integer. The displacement for LAY is treated as a 20-bit signed binary integer.

No storage references for operands take place, and the address is not inspected for access exceptions.

Condition Code: The code remains unchanged.

Program Exceptions:

- Operation (LAY if the long-displacement facility is not installed)

Programming Notes:

1. An example of the use of the LOAD ADDRESS instruction is given in Appendix A, "Number Representation and Instruction-Use Examples."
2. LOAD ADDRESS may be used to increment the rightmost bits of a general register, other than register 0, by the contents of the D_2 field of the instruction. LOAD ADDRESS (LAY) may also be used to decrement the rightmost bits of a register, other than register 0. The register to be incremented should be designated by R_1 and by either X_2 (with B_2 set to zero) or B_2 (with X_2 set to zero). The instruction updates 24 bits in the 24-bit addressing mode, 31 bits in the 31-bit addressing mode, and 64 bits in the 64-bit addressing mode.

LOAD ADDRESS EXTENDED

```
LAE    R1,D2(X2,B2)     [RX]

┌─────────┬────┬────┬────┬────────────┐
│  '51'   │ R1 │ X2 │ B2 │     D2     │
└─────────┴────┴────┴────┴────────────┘
0         8    12   16   20           31
```

The address specified by the X_2, B_2, and D_2 fields is placed in general register R_1. Access register R_1 is loaded with a value that depends on the current value of the address-space-control bits, bits 16 and 17 of the PSW. If the address-space-control bits are 01 binary, the value placed