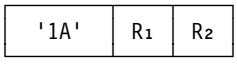
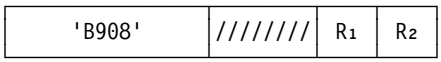
ADD

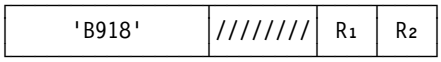
AR R1,R2 [RR]



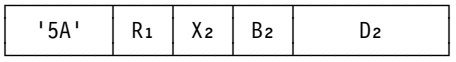
AGR R1,R2 [RRE]



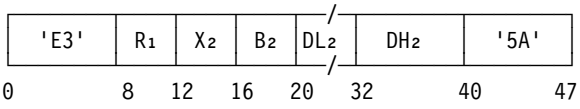
AGFR R1,R2 [RRE]



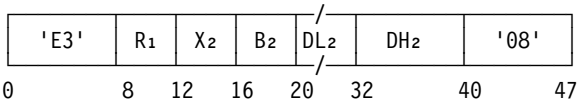
A R1,D2(X2,B2) [RX]



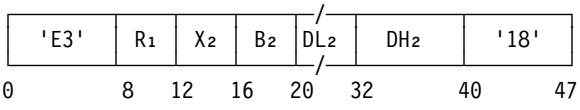
AY R1,D2(X2,B2) [RXY]



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



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



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:

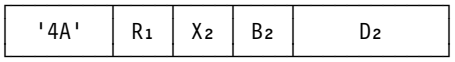
0 Result zero; no overflow
1 Result less than zero; no overflow
2 Result greater than zero; no overflow
3 Overflow

Program Exceptions:

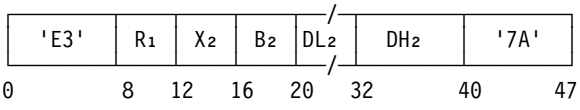
- Access (fetch, operand 2 of A, AY, AG, and AGF only)
- Fixed-point overflow
- Operation (AY, if the long-displacement facility is not installed)

ADD HALFWORD

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

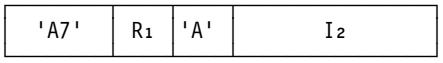


AHY R1,D2(X2,B2) [RXY]

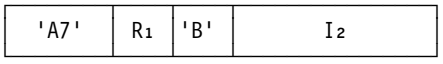


ADD HALFWORD IMMEDIATE

AHI R1,I2 [RI]



AGHI R1,I2 [RI]



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

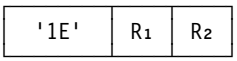
Program Exceptions:

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

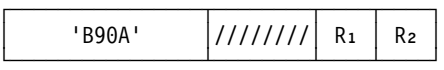
Programming Note: An example of the use of the ADD HALFWORD instruction is given in Appendix A, "Number Representation and Instruction-Use Examples."

ADD LOGICAL

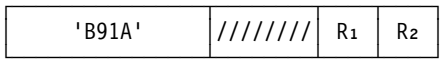
ALR R1,R2 [RR]



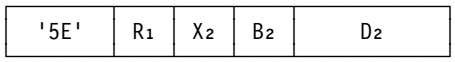
ALGR R1,R2 [RRE]



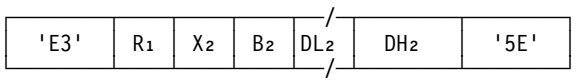
ALGFR R1,R2 [RRE]



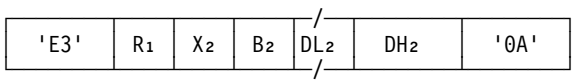
AL R1,D2(X2,B2) [RX]



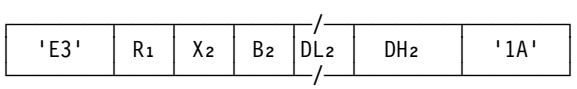
ALY R1,D2(X2,B2) [RXY]



ALG R1,D2(X2,B2) [RXY]



ALGF R1,D2(X2,B2) [RXY]



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:

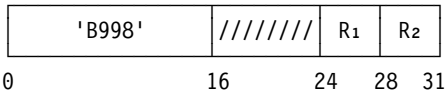
0 Result zero; no carry
1 Result not zero; no carry
2 Result zero; carry
3 Result not zero; carry

Program Exceptions:

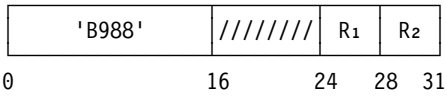
- Access (fetch, operand 2 of AL, ALY, ALG, and ALGF only)
- Operation (ALY, if the long-displacement facility is not installed)

ADD LOGICAL WITH CARRY

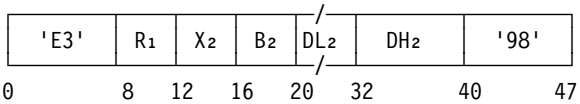
ALCR R1,R2 [RRE]



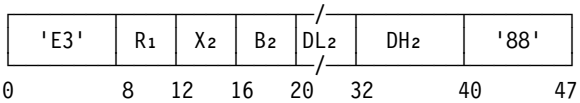
ALCGR R1,R2 [RRE]



ALC R1,D2(X2,B2) [RXY]



ALCG R1,D2(X2,B2) [RXY]



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:

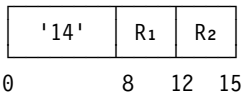
- Access (fetch, operand 2 of ALC and ALCG only)

Programming Notes:

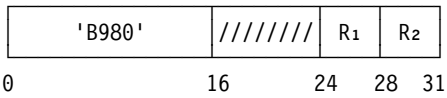
1. A carry is represented by a one value of bit 18 of the current PSW. Bit 18 is the leftmost bit of the two-bit condition code in the PSW. Bit 18 is set to one by an execution of an ADD LOGICAL or ADD LOGICAL WITH CARRY instruction that produces a carry out of bit position 0 of the result.
2. ADD and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

AND

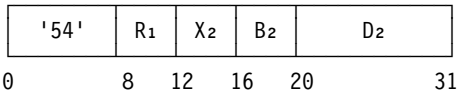
NR R1,R2 [RR]



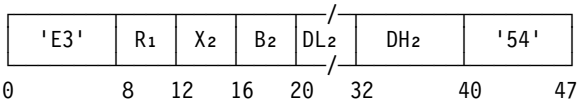
NGR R1,R2 [RRE]



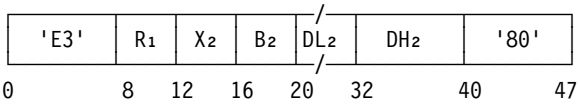
N R1,D2(X2,B2) [RX]



NY R1,D2(X2,B2) [RXY]



NG R1,D2(X2,B2) [RXY]



NI D1(B1),I2 [SI]