Richard Cebula – HLASM

Introduction to Assembler Programming Cheat Sheet

Introduction to Assembler Programming – Cheat Sheet

- LOAD / STORE Instructions
- MOVE Instructions
- Logical Instructions
- Defining Data and Literals
- Branch Instructions
- Arithmetic Instructions
- BRANCH ON COUNT (Looping)

LOAD / STORE Instructions

- LOAD data from storage to a register

```
L    1,NUMBER     LOAD REGISTER 1 WITH NUMBER (32-BITS)
LH   1,NUMBER     LOAD REGISTER 1 WITH NUMBER (16-BITS)
LG   1,NUMBER     LOAD REGISTER 1 WITH NUMBER (64-BITS)
LR   1,2          LOAD REGISTER 1 WITH REGISTER 2 (32-BITS)
LGR  1,2          LOAD REGISTER 1 WITH REGISTER 2 (64-BITS)
IC   NUM1,NUM2    INSERT CHARACTER (1 BYTE)
```

- STORE data from a register to storage

```
ST   1,NUMBER     STORE REGISTER 1 TO NUMBER (32-BITS)
STH  1,NUMBER     STORE REGISTER 1 TO NUMBER (16-BITS)
STG  1,NUMBER     STORE REGISTER 1 TO NUMBER (64-BITS)
STC  1,NUMBER     STORE REGISTER 1 TO NUMBER (1 BYTE)
```

MOVE Instructions

- MOVE Data from storage to storage
  - Move SOURCE to TARGET – implied length of move is length of TARGET

    ```
    MVC  TARGET,SOURCE
    ```

  - Move SOURCE to TARGET – length of move is LENGTH

    ```
    MVC  TARGET(LENGTH),SOURCE
    ```

  - Move SOURCE to TARGET (with displacements) – length of move is LENGTH

    ```
    MVC  TARGDISP(LENGTH,TARG),DISP(SRC)
    ```

Logical Instructions – Result always ends up in 1st operand

- AND

```
N    1,NUMBER     AND REGISTER 1 WITH NUMBER (32-BITS)
NG   1,NUMBER     AND REGISTER 1 WITH NUMBER (64-BITS)
NR   1,2          AND REGISTER 1 WITH REGISTER 2 (32-BITS)
NGR  1,2          AND REGISTER 1 WITH REGISTER 2 (64-BITS)
NC   NUM1,NUM2    AND NUM1 WITH NUM2 (UP TO 256-BYTES)
```

- OR

```
O    1,NUMBER     OR REGISTER 1 WITH NUMBER (32-BITS)
OG   1,NUMBER     OR REGISTER 1 WITH NUMBER (64-BITS)
OR   1,2          OR REGISTER 1 WITH REGISTER 2 (32-BITS)
OGR  1,2          OR REGISTER 1 WITH REGISTER 2 (64-BITS)
OC   NUM1,NUM2    OR NUM1 WITH NUM2 (UP TO 256-BYTES)
```

- EXCLUSIVE OR

```
X    1,NUMBER     XOR REGISTER 1 WITH NUMBER (32-BITS)
XG   1,NUMBER     XOR REGISTER 1 WITH NUMBER (64-BITS)
XR   1,2          XOR REGISTER 1 WITH REGISTER 2 (32-BITS)
XGR  1,2          XOR REGISTER 1 WITH REGISTER 2 (64-BITS)
XC   NUM1,NUM2    XOR NUM1 WITH NUM2 (UP TO 256-BYTES)
```

Defining Data

- Define MYNUMBER as a fullword (32-bits) with initial value 0 – note that the label MYNUMBER must start in column 1

```
MYNUMBER DC    F'0'
```

- Define MYNAME as a series of bytes with length 20 and initial value “hello” (space padded)

```
MYNAME   DC    CL20'hello'
```

- Define MYNUMBER as 4 halfwords (16-bits each) with initial value 12 (each)

```
MYNUMBER DC    4H'12'
```

- Define MYTOTAL as some uninitialised storage reserving a fullword (32-bits) for it

```
MYNUMBER DS    F
```

- Use a literal in an instruction to load value 4097 into register 1

```
L        1,=F'4097'
```

BRANCH Instructions

```
L    1,NUM1                             LOAD VALUE NUM1 TO REG 1
L    2,NUM2                             LOAD VALUE NUM2 TO REG 2
CR   1,2                                COMPARE THE REGISTERS
BL   ONE_LESS              BRANCH IF REGISTER 1 < REGISTER 2
BG   ONE_MORE              BRANCH IF REGISTER 1 > REGISTER 2
BE   ONE_EQUAL_TWO         BRANCH IF REGISTER 1 = REGISTER 2
```

Arithmetic Instructions – Result *normally* ends up in 1st operand

- ADD and SUBTRACT (Condition code is updated)

```
A    1,NUM    ADD REGISTER 1 WITH NUM (32-BITS, SIGNED)
AR   1,2      AND REGISTER 1 WITH REGISTER 2 (32-BITS, SIGNED)
SL   1,NUM    REGISTER_1 = REGISTER_1 - NUM (32-BITS, LOGICAL)
SLR 1,2       REGISTER_1 = REGISTER_1 - REGISTER_2
```

- MULTIPLY (CC = unchanged)

```
MR   2,7      MULTIPLY 32-bits in REGISTER 3 by REGISTER 7
Result is stored as:
   top 32-bits are in register 2
   bottom 32-bits are in register 3
```

- DIVIDE (CC = unchanged)

```
DR   2,7     DIVIDE 64-BITS IN REGISTERS 2 AND 3 BY REGISTER 7
Top 32-bits are in register 2, bottom 32-bits in register 3
Result is stored as:
   Quotient is stored in register 3
   Remainder is stored in register 2
```

BRANCH ON COUNT

```
* SAY “HELLO WORLD” 10 TIMES
          L       1,LOOP_MAX     LOAD LOOP_MAX(VALUE 10) INTO REG 1
LOOP      DS      0H             START OF THE LOOP
          WTO     'HELLO WORLD'  SAY HELLO WORLD
          BCT     1,LOOP         BRANCH TO LOOP IF REG 1 > 0
```