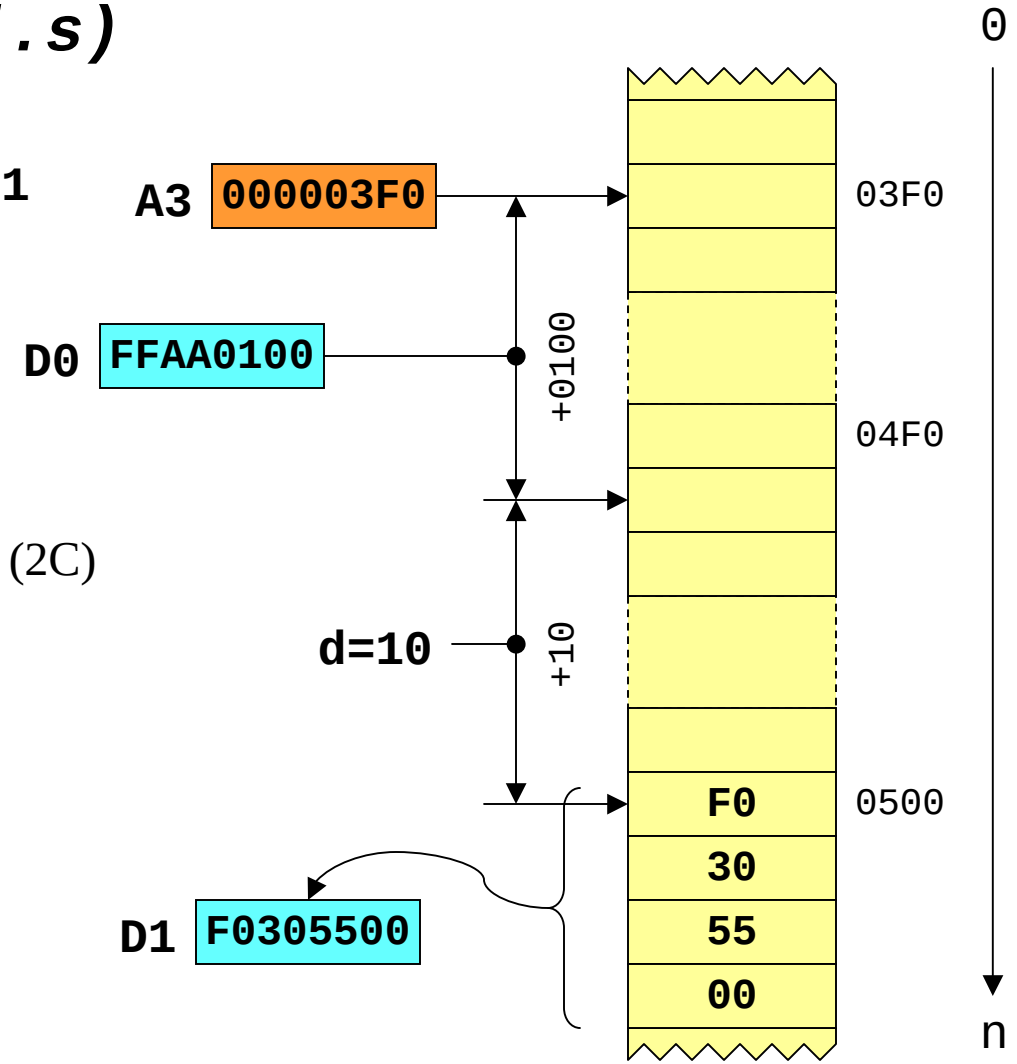
Register Indirect with Index (& Displacement) Addressing

d_8(Ai,Xj.s) : mem[d_8+Ai+Xj.s]

(d_8,Ai,Xj.s)

MOVE.L $10(A3,D0.W),D1



both index & d can be nagative (2C)

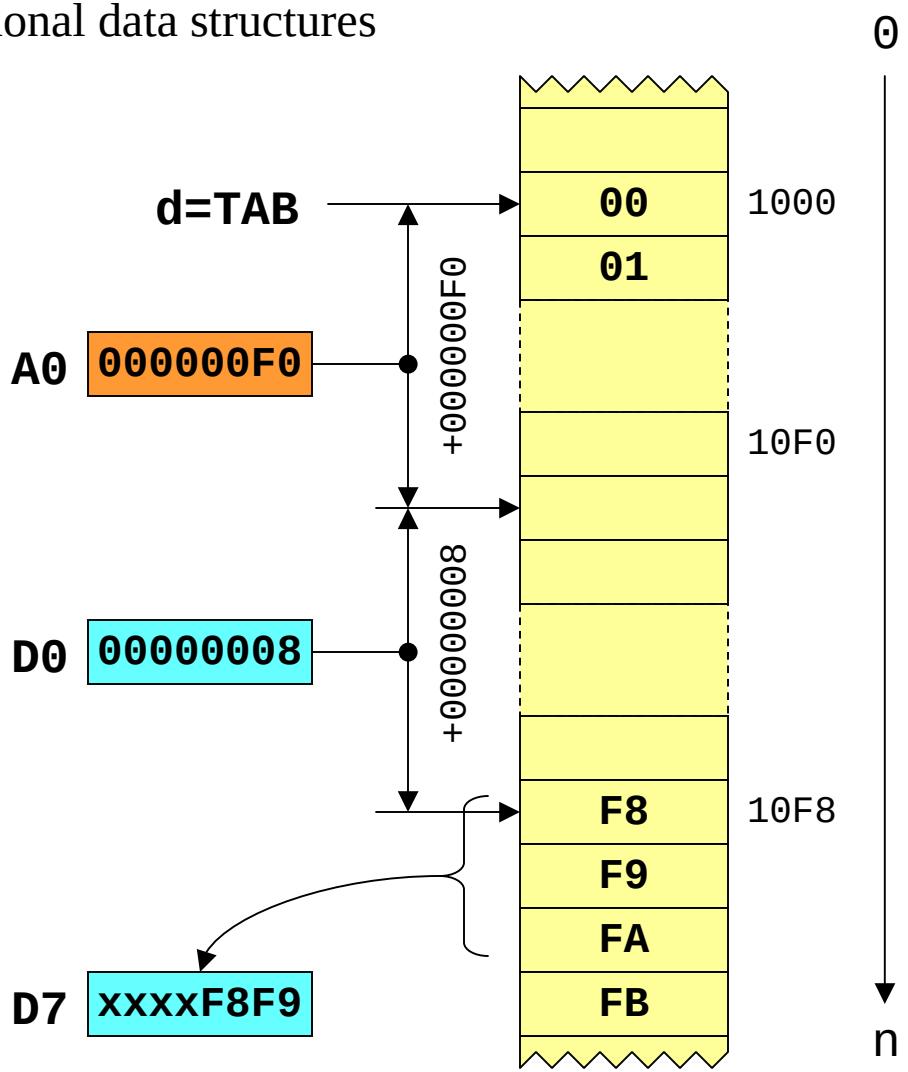
watch for index size (2C)

only 8-bit d (2C): -128...+127

Register Indirect with Index (& Displacement) Addressing

access to 2-dimensional data structures

```
        ...
        MOVE.W TAB(A0,D0.L),D7
        ...
        ...
        ...
TAB     DC.B   0,1,2,3...
```



68000 – 8-bit displacement,
d: first & last 127B of memory

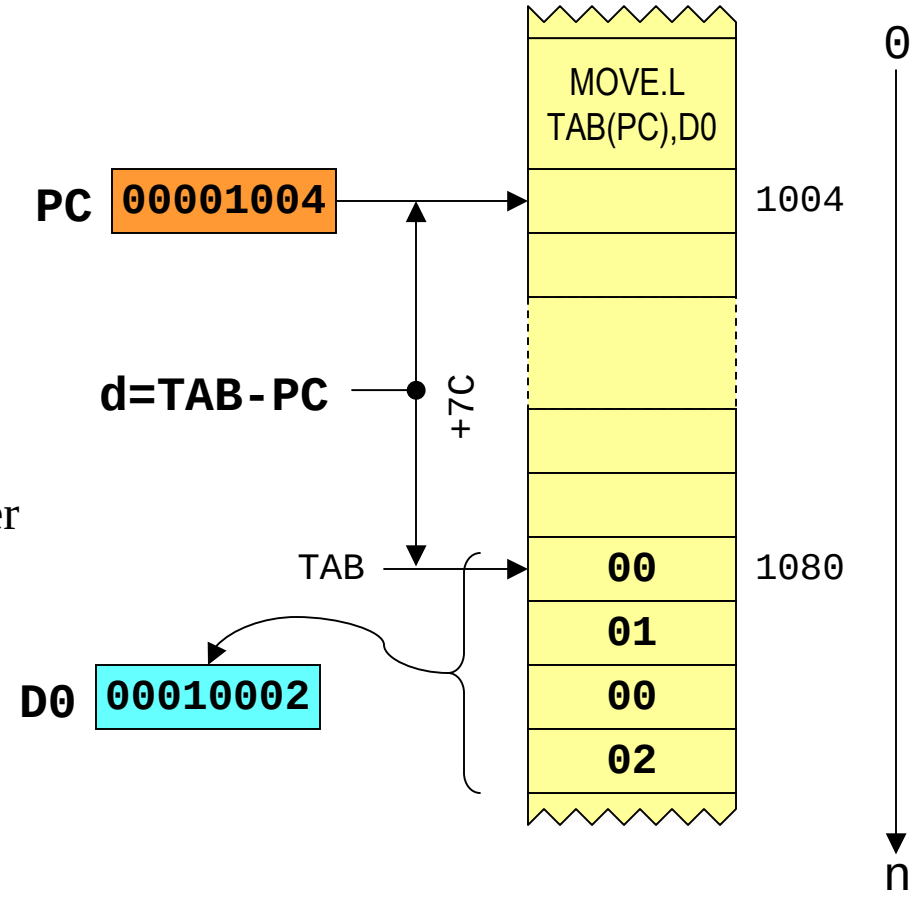
68020+ - any displacement size,
8,16,32 bit – any data location

Program Counter Relative Addressing Modes

$d_{16}(PC)$: $mem[d_{16}+PC]$

$d_8(PC,Xi.s)$: $mem[d_8+PC+Xi.s]$

```
        MOVE.L TAB(PC),D0
        ...
        ...
TAB     DC.W   1,2
```



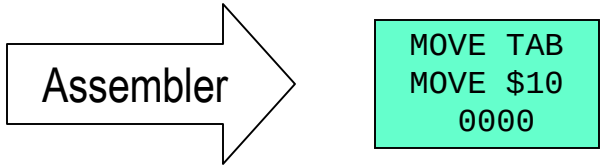
displacement is calculated by assembler

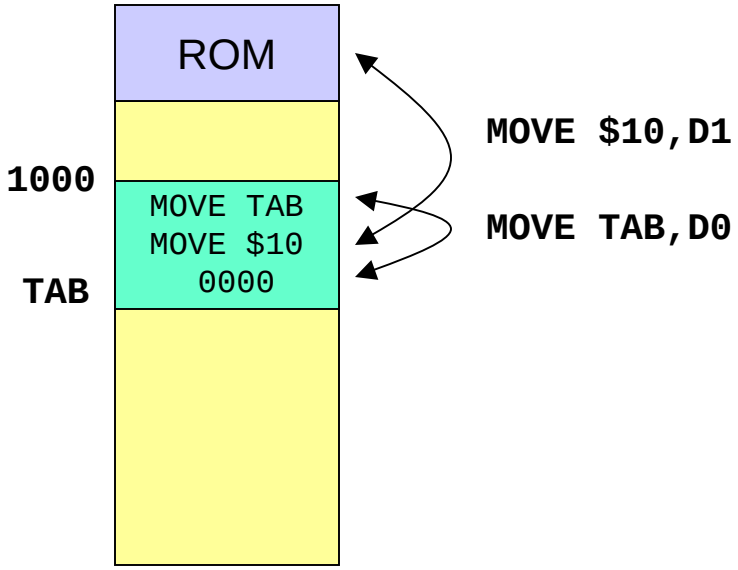
only for source operands

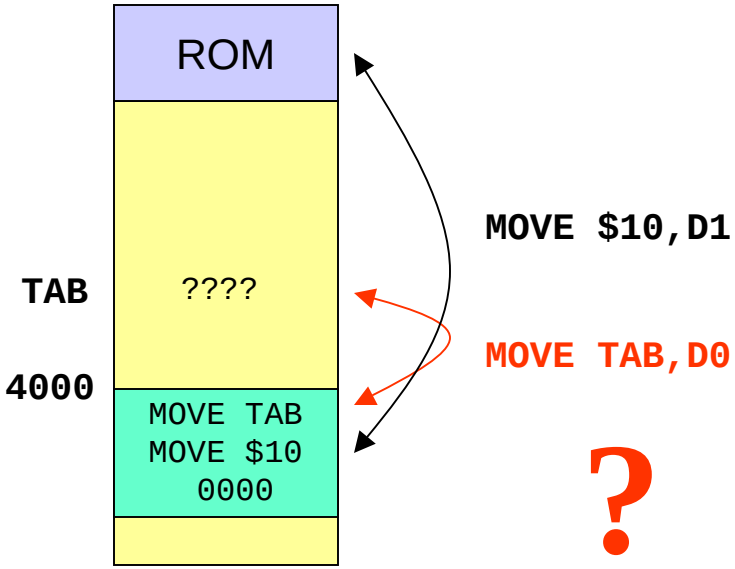
~~`MOVE.L D0,TAB(PC)`~~

Program relocation

```
        ORG $1000
        ...
        MOVE TAB,D0
        ...
        MOVE $10,D1
        ...
TAB     DS.L    0
```

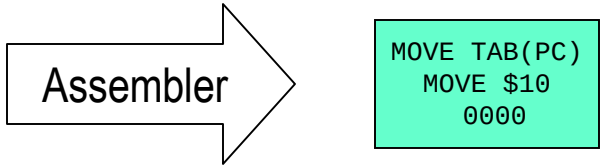


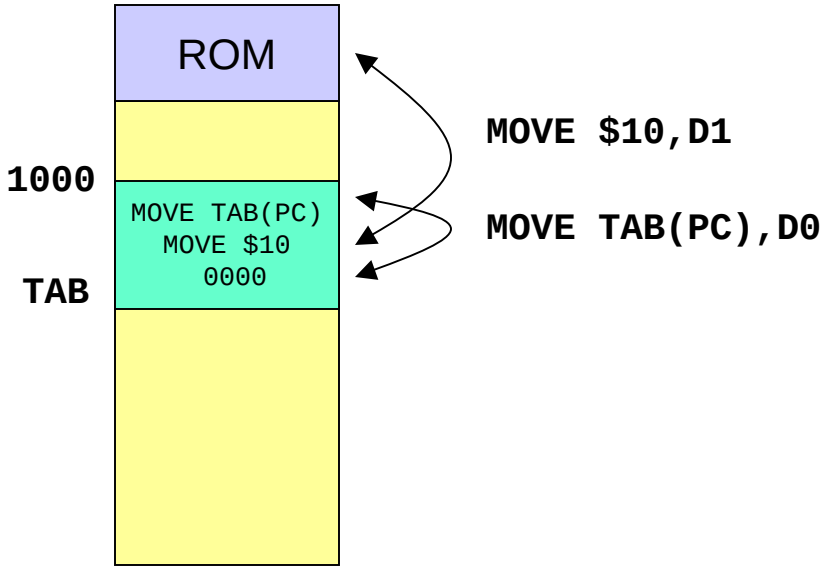


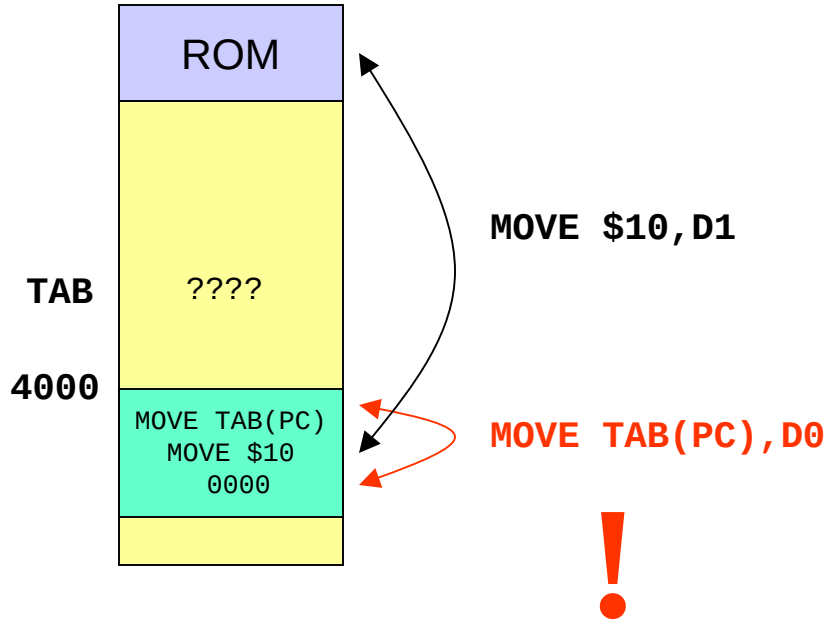


Program relocation

```
        ORG $1000
        ...
        MOVE TAB(PC),D0
        ...
        MOVE $10,D1
        ...
TAB     DS.L    0
```







Program relocation

A relocatable program is assembled assuming a fix starting location.

All other instructions and data areas are assembled relative to this base.

When the program is loaded, the relocatable operands are adjusted in run-time to correspond with the actual locations assigned by the loader.

Relocateble programs techniques
- PC relative addressing modes
- PC relative jumps
- PC relative subroutine calling