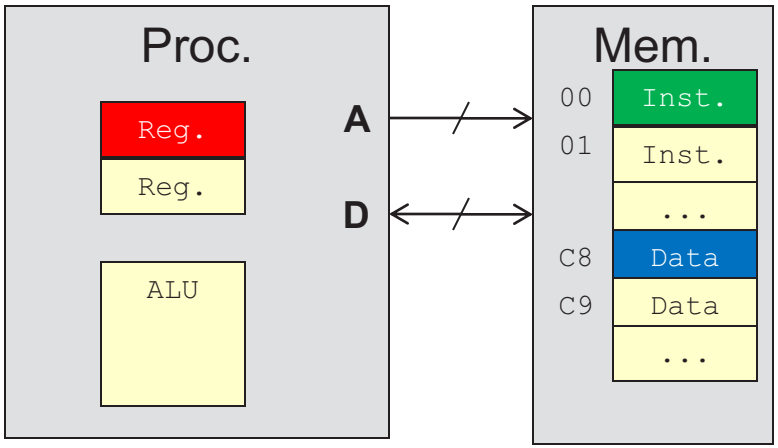
Coldfire Instruction Classes

- Data Transfer
  - Move data between processor & memory
  - Move data between registers w/in processor
  - Can specify .B, .W, .L size
- ALU
  - Performs arithmetic and logic operations
  - Only .L size => Ops. must be on a full 32-bit longword contents
- Control / Program Flow
  - Unconditional/Conditional Branch
  - Subroutine Calls
- Privileged / System Instructions
  - Instructions that can only be used by OS or other “supervisor” software (e.g. STOP, certain HW access instructions, etc.)

Operand Locations

- In almost all instruction sets, operands can be…
  - A register value (e.g. D0)
  - A value in a memory location (e.g. value at address 0xC8)
  - A constant stored in the instruction itself (known as an 'immediate' value) [e.g. ADDI #1,D0]
- Thus, our instructions must be able to specify which of these three locations is where the operand is located

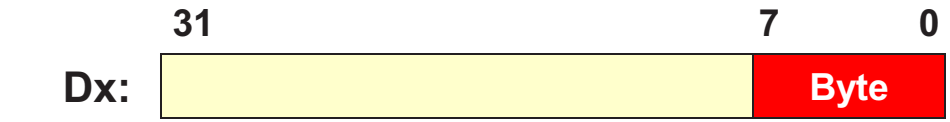


Data Transfer Instruction

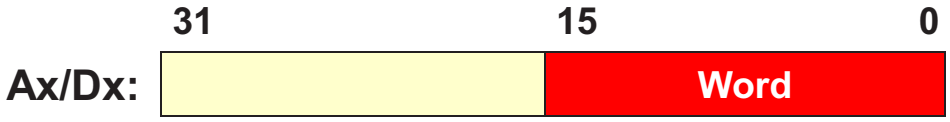
- MOVE.s src,dst
  - .s = .B, .W, .L
  - Copies src operand to dst operand
  - SRC operand specifies data in a:
    - Reg. (e.g. D0 or A4)
    - Mem. = Specified with the address of desired source location
    - Immediate = Constant (preceded w/ '#' sign in assembly instruc.)
  - DST operand specifies a location to put the source data & can be a:
    - Reg.
    - Mem. = Specified with the address of desired destination location
- Examples
  - MOVE.B D0,0x1C ; 0x = hex modifier
    - Moves the byte from reg. D0 to memory byte @ address 0x1c
  - MOVE.W #0x1C,D0 ; # = Immediate (no # = address/mem. oprnd.)
    - Moves the constant 001C hex to D0

Registers & Data Size

- Most CF instructions specify what size of data to operate on
  - MOVE.B
  - MOVE.W
  - MOVE.L
- Register sizes are right-justified (start at LSB and work left)



Byte operations only access bits 7-0 of a register (upper bits are left alone)



Word operations only access bits 15-0 of a register (upper bits are left alone)

Longword operations use the entire 32-bit value

Examples

- Initial Conditions:

D0: 1 2 3 4 5 6 7 8

D1: A A A A B B B B

- MOVE.B D0,D1

D1: A A A A B B 7 8

- MOVE.W #0xFEDC,D1

D1: A A A A F E D C

immediate (constant) ↑ ↑ base hex

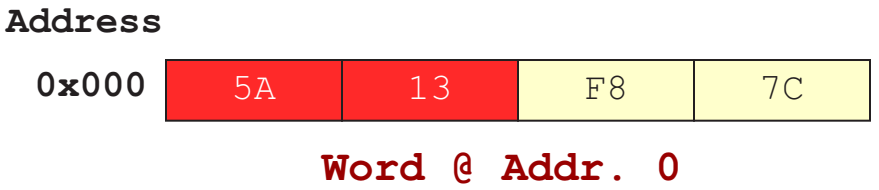
- MOVE.L D0,D1

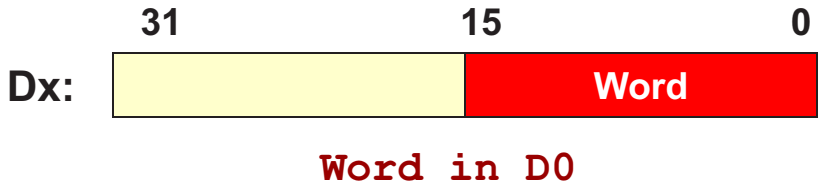
D1: 1 2 3 4 5 6 7 8

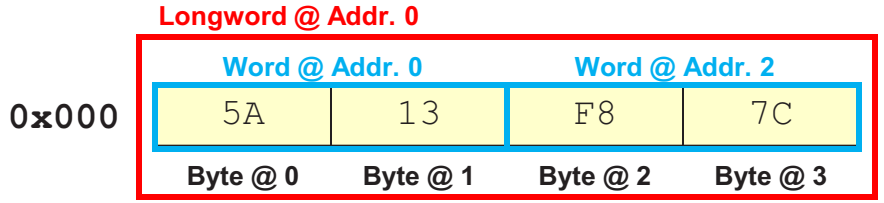
Memory & Data Size

Recall…

- Memory operands are addressed by their starting address and size
  - Whereas registers operands always start with LSB and get appropriate size data, memory operands get the appropriate sized data from starting address and continuing towards larger addresses
  - This makes it look left justified due to the big-endian ordering
- Valid words start @ addresses that is multiples of 2
- Valid longwords start @ addresses that are multiples of 4







Examples

- Initial Conditions:

000000	12 34 56 78
000004	9A BC DE F0

- MOVE.B 1,2

no '#' so treat as address

000000	12 34 **34** 78
000004	9A BC DE F0

- MOVE.W 0x2,0x4

000000	12 34 56 78
000004	**56 78** DE F0

- MOVE.L #$FEDCBA98,4

immediate — another hex indicator — address

000000	12 34 56 78
000004	**FE DC BA 98**

Examples

- Initial Conditions:

000000	12 34 56 78
000004	9A BC DE F0
D0:	1 2 3 4 5 6 7 8

- MOVE.B 0x5,D0

D0:	1 2 3 4 5 6 **B C**

- MOVE.W 4,D0

D0:	1 2 3 4 **9 A B C**

- MOVE.L #3,4

000000	12 34 56 78
000004	**00 00 00 03**