DCD 32 bit (full word) '' : single char, DCD, DCW, DCB

DCW 16 bit (half word)

DCB 8 bit " " : string, DCB only

ALIGN align to a divisible by 4 add.

= same as DCB. i.e. x = 0x41

& DCD x & 0x123456

% SPACE allocate and zero a space.

0x/& hex e.g. MOV R1, #&9C

DCD &9C

2_ binary e.g. DCD 2_100011100

8_ octal e.g. MOV R1, #8_234

no prefix decima

```
    AREA PROG1, code, readonly        R0          PC          RTL
    ENTRY
0   LDR R0, [R1]                     0xE5910000  0x4         [R0] ← [[R1]].
4   LDR R0, 0xFF                     0xFF        0x8         [R0] ← 0xFF
C   LDR R0, 0xFFF                    0xFFF       0xC         [R0] ← 0xFFF
10  LDR R0, X; pseudo-instructions   0xAAAAAAAA  0x10        [R0] ← [X]
14  LDR R0, =X; pseudo-instructions  0x1C        0x14        [R0] ← X
                (load x's address)
18  ADR R0, X; pseudo-instructions   0x1C        0x18        [R0] ← X
                (load add.)
Loop B Loop
1C
X   DCD 0xAAAAAAAA
    END
```

test.s

```
1     AREA prog1, code, READONLY
2  ENTRY
3     MOV r0, #125
4     LDR r1,[r0]
5     LDR r2, =0xFF ; pseudo-instruction
6     LDR r3, =0xFFF ; pseudo-instruction
7     LDR r4, X ; pseudo-instruction
8     LDR r5, =X ; pseudo-instruction
9     ADR r6, X ; pseudo-instruction
10 loop B loop
11 X DCD 0xAAAAAAAA
12    END
13
```

Memory 1

Address: 0

```
0x00000000: E3 A0 00 7D
0x00000004: E5 91 10 00
0x00000008: E3 A0 20 FF
0x0000000C: E5 9F 30 10
0x00000010: E5 9F 40 08
0x00000014: E5 9F 50 0C
0x00000018: E2 8F 60 00
0x0000001C: EA FF FF FE
0x00000020: AA AA AA AA
0x00000024: 00 00 0F FF
0x00000028: 00 00 00 20
0x0000002C: 00 00 00 00
0x00000030: 00 00 00 00
0x00000034: 00 00 00 00
```

Registers

Register	Value
Current	
R0	0x0000007D
R1	0xE3A0007D
R2	0x000000FF
R3	0x00000FFF
R4	0xAAAAAAAA
R5	0x00000020
R6	0x00000020
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000001C
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Inte...	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x0000001C
Mode	Supervisor
States	117
Sec	0.00000000

125
[[R0]]

test.s

```
1     AREA More_data_definitions, CODE, READONLY
2     ENTRY
3     MOV r0, # 0xFC  ;Store a Positive HEX number in r0
4     MOV r1, #-0xFC  ;Store a negative HEX number in r1
5     MOV r2, # 240   ;Store a Positive decimal number in r2
6     MOV r3, # -240  ;Store a negative decimal number in r3
7  loop B loop
8  one = 1,1,1,1      ;the "=" here means DCB
9  Letter DCB &41     ;the "&" here means an ASCII code in HEX (MUST BE between 00 and FF)
10                    ;The "0x" prefix is NOT allowed after the "&"
11                    ;DCB can start at any memory location.
12 two DCW 2          ;Must start at an even address location.
13                    ;One byte to be skipped to adjust the location counter.
14                    ;IF YOU PUT ALIGN BEFORE THIS DCW, IT WILL SKIP 3 BYTES, NOT JUST ONE,
15                    ;TO MAKE THE ADDRESS DIVISIBLE BY 4
16 four & 4,4         ;the "&" here means DCD
17                    ;DCD must start at a divisible by 4 address location
18    DCD 2_1010          ;Binary positive number
19    DCD -2_1010         ;Binary negative number
20    DCD 8_12345670      ;Octal positive number
21    DCD -8_12345670     ;Octal negative number
22    DCB 1               ;Any data directive can be without label
23 data_1 SPACE 5     ;reserves a ZEROED 5 bytes block of memory
24 data_2 % 5         ;the "%" here means SPACE
25    ALIGN           ;ADVANCE THE LOCATION COUNTER TO THE NEXT DIVISIBLE BY 4 ADDRESS LOCATION
26 data_3 SPACE 5
27    DCB "AAAAAAA"
28    END
```

Registers

Register	Value	
Current		
R0	0x000000FC	
R1	0xFFFFFF04	-0xFC
R2	0x000000F0	240
R3	0xFFFFFF10	-240
R4	0x00000000	
R5	0x00000000	
R6	0x00000000	
R7	0x00000000	
R8	0x00000000	
R9	0x00000000	
R10	0x00000000	
R11	0x00000000	
R12	0x00000000	
R13 (SP)	0x00000000	
R14 (LR)	0x00000000	
R15 (PC)	0x00000010	4 lines executed
CPSR	0x000000D3	
SPSR	0x00000000	
User/System		
Fast Inte...		
Interrupt		
Supervisor		
Abort		
Undefined		
Internal		
PC $	0x00000010	
Mode	Supervisor	
States	10	
Sec	0.00000000	

Memory 1

Address: 0

```
0x00000000: E3 A0 00 FC
0x00000004: E3 E0 10 FB
0x00000008: E3 A0 20 F0
0x0000000C: E3 E0 30 EF
0x00000010: EA FF FF FE
0x00000014: 01 01 01 01
0x00000018: 41 00 00 02
0x0000001C: 00 00 00 04
0x00000020: 00 00 00 04
0x00000024: 00 00 00 0A
0x00000028: FF FF FF F6
0x0000002C: 00 29 CB B8
0x00000030: FF D6 34 48
0x00000034: 01 00 00 00
0x00000038: 00 00 00 00
0x0000003C: 00 00 00 00
0x00000040: 00 00 00 00
0x00000044: 00 41 41 41
0x00000048: 41 41 41 41
0x0000004C: 00 00 00 00
0x00000050: 00 00 00 00
```

MOV r0, #0xFC
MOV r1, #-0xFC
MOV r2, 240
MOV r3, -240
loop B loop
DCB 1,1,1,1
DCB 0x41 DCW 2
→ DCW would only start at half/full word
DCD 4,4
DCD 2_1010
DCD -2_1010
DCD 8_12345670
DCD -8_12345670
DCB 1
data_1 SPACE 5
data_2 SPACE 5
→ vacant due to ALIGN
data_3 SPACE 5
"AAAAAAA"

Shifts:

Static:

logical: L\R: implement 0 in vacant position

arthimetic: L: same as LSL

R: replicate sign bits

circular L\R: the bit shift-out in one end shift-in in another end

Circular with carry: the carry bit is in circular too.

* the sign bit is in circular in both ROR/RRX.

LSL: #0 ~ #31 ASL is implemented in LSL.

LSR: #1 ~ #32 (LSR#0 is same as LSL#0).

ASR: #1 ~ #32

ROR: #1 ~ #31 (ROL #n = ROR #32-n).

RRX shift one position per time

(ROR + a shift of #0 -> RRX).

dynamic: the number change in runtime.

if the number of dynamic shift ≥ 32, zero will store in destination.

* ARM has NO explicit shift operation. All shifts are done with other data processing operations.