

These slides are being provided with permission from the copyright for in-class (CS2208B) use only. The slides must not be reproduced or provided to anyone outside of the class.

All download copies of the slides and/or lecture recordings are for personal use only. Students must destroy these copies within 30 days after receipt of final course evaluations.

Tutorial 08:

ARM Data Definition Directives

Computer Science Department

CS2208: Introduction to Computer Organization and Architecture

Winter 2020-2021

Instructor: Mahmoud R. El-Sakka

Office: MC-419

Email: elsakka@csd.uwo.ca

Phone: 519-661-2111 x86996

ARM Assembly Directives

❑ Assembly language directives include:

AREA To name a region of **code** or **data**

ENTRY The execution starting point

END The physical end of the program

name EQU *v. expr* Equate a *name* to the *value* of the *v. expr*
Will not make any memory allocation, i.e., Similar to #define in C

{label} DCD *v. expr* {, *v. expr* } ... Set up one or more **32-bit constant** in memory
Must start at a multiple of 4 address location

{label} DCW *v. expr* {, *v. expr* } ... Set up one or more **16-bit constant** in memory
Must start at an even address location

{label} DCB *v. expr* {, *v. expr* } ... Set up one or more **8-bit constant** in memory
Can start anywhere

{label} SPACE *size expr* Reserves a zeroed block of memory
Can start anywhere

ALIGN Ensures that next instruction is correctly aligned on 32-bit boundaries, i.e., to start at a multiple of 4 address location

ARM Assembly Directives

- ❑ The “**v. expr**” can be any *constant-value expression*, i.e., its value MUST be evaluated during assembly phase, not during execution.
- ❑ The “**v. expr**” examples:
 - $2*50/3 \rightarrow$ to be evaluated to 0x21 (i.e., 33)
 - ‘A’ \rightarrow to be evaluated to 0x41 (i.e., 65)
 - “ABC” \rightarrow to be evaluated to 0x414243

The single quotation for a single character only.
It can be used with **DCB**, **DCW** or **DCD**

The double quotation for a string.
It MUST be used with **DCB**

ARM Assembly Directives

- ❑ Some symbols in Keil assembler have different meanings, based on their location within the instruction:

○ Equal sign “=”

- at the opcode column *means* DCB
- as a prefix to the 2nd operand of an LDR instruction *means* pseudo instruction

Example 1:

XYZ = 0x41 ; the = sign in this context means DCB, i.e.,

XYZ DCB 0x41

What will happen if the “=” sign is omitted?

Example 2:

LDR r0,=0x12345678 ; to LDR the 32-bit value 0x12345678 into r0

LDR r0,=PPP ; to LDR the 32-bit address of PPP into r0

the = sign in this context means the LDR here is a pseudo instruction

○ Ampersand sign “&”

- at the opcode column *means* DCD
- as a prefix to an operand *means* a HEX value (i.e., similar to 0x)

Example 3:

AAA & 0x123456 ; the & sign in this context means DCD, i.e.,

AAA DCD 0x123456

Example 4:

MOV r0,#&8F ; the & sign in this context means a HEX value

○ Percent sign “%”

- at the opcode column *means* SPACE

Example 5:

BBB % 0x40 ; the % sign in this context means SPACE, i.e.,

BBB SPACE 0x40

Writing Numbers with Various Radix

□ The Keil assembler uses

- a prefix **0x** or **&** to indicate **hexadecimal** constant, e.g.,

```
MOV r1, #0x9C
```

```
MOV r1, #&9C
```

or

```
DCD 0x9C
```

```
DCD &9C
```

- a prefix **2_** to indicate **binary** constant, e.g.,

```
MOV r1, #2_10011100
```

or

```
DCD 2_10011100
```

- a prefix **8_** to indicate **octal** constant, e.g.,

```
MOV r1, #8_234
```

or

```
DCD 8_234
```

- **no** prefix to indicate **decimal** constant, e.g.,

```
MOV r1, #156
```

or

```
DCD 156
```

In ARM assembly,
the "#" means
Literal or immediate
addressing mode

In ARM assembly,
It is **illegal** to use "#" with
DCD, DCW, or DCB

Data Definition Directives

```
AREA More_data_definitions, CODE, READONLY
```

```
ENTRY
```

```
MOV r0, # 0xFC; Store a Positive HEX number in r0
```

```
MOV r1, #-0xFC; Store a negative HEX number in r1
```

```
MOV r2, # 240; Store a Positive decimal number in r2
```

```
MOV r3, # -240; Store a negative decimal number in r3
```

```
loop B loop
```

```
one = 1,1,1,1 ; the "=" here means DCB
```

```
Letter DCB &41 ; the "&" here means an ASCII code in HEX (MUST BE between 00 and FF)
```

```
; The "0x" prefix is NOT allowed after the "&"
```

```
; DCB can start at any memory location.
```

```
two DCW 2 ; Must start at an even address location.
```

```
; One byte to be skipped to adjust the location counter.
```

```
; IF YOU PUT ALIGN BEFORE THIS DCW, IT WILL SKIP 3 BYTES, NOT JUST ONE,
```

```
; TO MAKE THE ADDRESS MULTIPLE OF 4
```

```
four & 4,4 ; the "&" here means DCD
```

```
; DCD must start at a multiple of 4 address location
```

```
DCD 2_1010 ; Binary positive number
```

```
DCD -2_1010 ; Binary negative number
```

```
DCD 8_12345670 ; Octal positive number
```

```
DCD -8_12345670 ; Octal negative number
```

```
DCB 1 ; Any data directive can be without label
```

```
data_1 SPACE 5 ; reserves a ZEROED 5 bytes block of memory
```

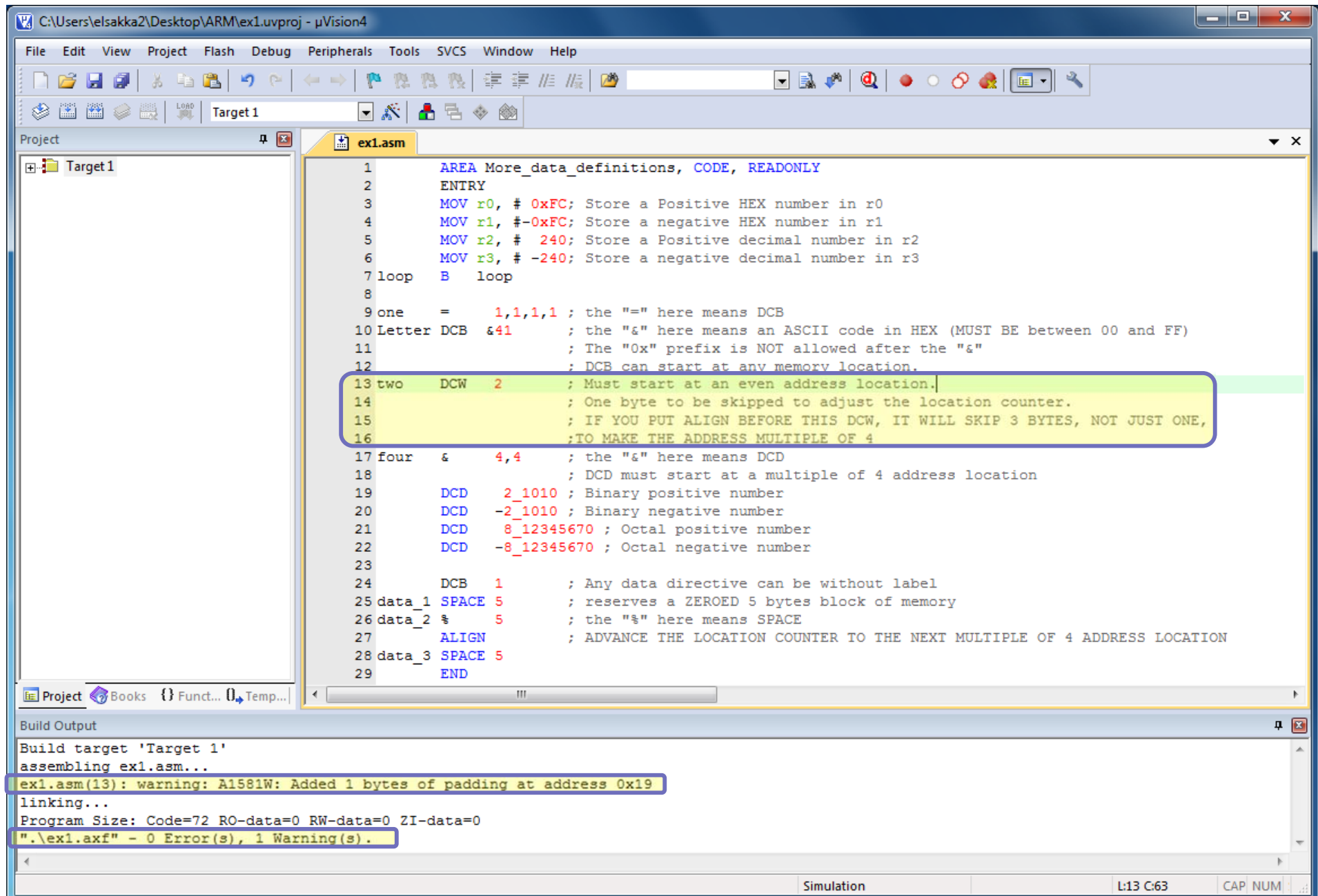
```
data_2 % 5 ; the "%" here means SPACE
```

```
ALIGN ; ADVANCE THE LOCATION COUNTER TO THE NEXT MULTIPLE OF 4 ADDRESS LOCATION
```

```
data_3 SPACE 5
```

```
END
```

Data Definition Directives



The screenshot shows the uVision4 IDE with the following assembly code in `ex1.asm`:

```

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

```

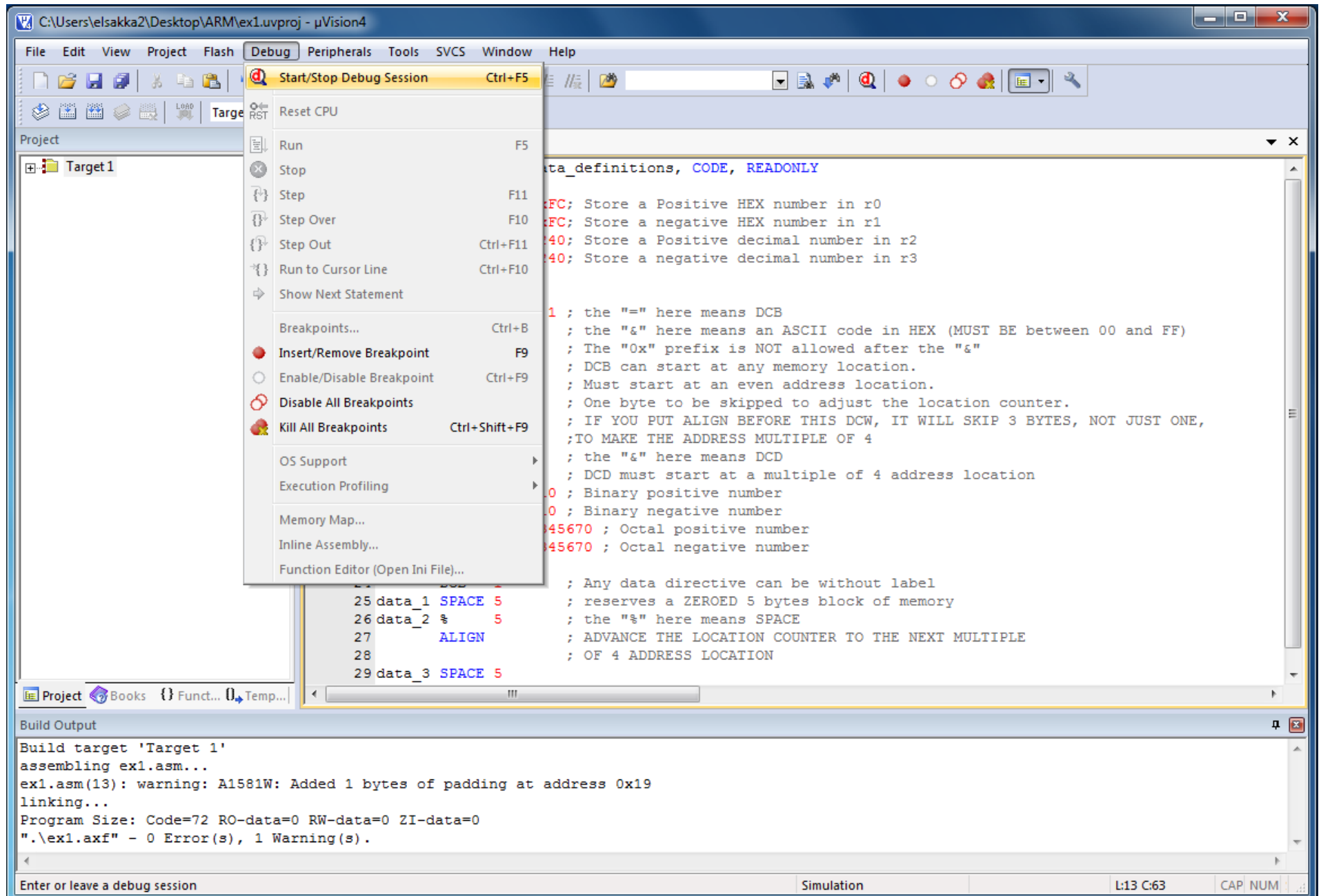
The build output at the bottom shows the following messages:

```

Build target 'Target 1'
assembling ex1.asm...
ex1.asm(13): warning: A1581W: Added 1 bytes of padding at address 0x19
linking...
Program Size: Code=72 RO-data=0 RW-data=0 ZI-data=0
".\ex1.axf" - 0 Error(s), 1 Warning(s).

```

Data Definition Directives



Data Definition Directives

The screenshot displays the uVision4 IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The left pane shows the 'Registers' window with a list of registers (R0-R15, CPSR, SPSR) and their current values. The main window is split into two panes. The top pane shows the disassembly of assembly code, with the following instructions highlighted:

```

3:      MOV r0, # 0xFC; Store a Positive HEX number in r0
0x00000000 E3A000FC MOV     R0,#0x000000FC
4:      MOV r1, #-0xFC; Store a negative HEX number in r1
0x00000004 E3E010FB MVN     R1,#0x000000FB
5:      MOV r2, # 240; Store a Positive decimal number in r2
0x00000008 E3A020F0 MOV     R2,#0x000000F0

```

The bottom pane shows the source assembly file 'ex1.asm' with the following code:

```

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
9 one  = 1,1,1,1 ; the "=" here means DCB
10 Letter DCB &41 ; the "&" here means an ASCII code in HEX (MUST BE between 00 and FF)
11 ; The "0x" prefix is NOT allowed after the "&"
12 ; DCB can start at any memory location.
13 two  DCW 2 ; Must start at an even address location.
14 ; One byte to be skipped to adjust the location counter.
15 ; IF YOU PUT ALIGN BEFORE THIS DCW, IT WILL SKIP 3 BYTES, NOT JUST ONE,
16 ; TO MAKE THE ADDRESS MULTIPLE OF 4
17 four & 4,4 ; the "&" here means DCD
18 ; DCD must start at a multiple of 4 address location
19 DCD 2_1010 ; Binary positive number

```

The bottom status bar shows 'Simulation' mode, 't1: 0.00000000 sec', 'L3 C:1', and 'CAP NUM'.

Data Definition Directives

The screenshot shows the uVision4 IDE interface. The main window displays assembly code for a file named `ex1.asm`. The code includes several `MOV` instructions and data definition directives like `AREA`, `ENTRY`, `DCB`, and `DCD`. The left sidebar shows the Project Window with the 'Memory Windows' menu open, listing Memory 1 through Memory 4. The bottom status bar indicates the simulation is running.

Assembly Code (ex1.asm):

```

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
9 one = 1,1,1,1 ; the "=" here means DCB
10 &41 ; the "&" here means an ASCII code in HEX (MUST BE between 00 and FF)
11 ; The "0x" prefix is NOT allowed after the "&"
12 ; DCB can start at any memory location.
13 ; Must start at an even address location.
14 ; One byte to be skipped to adjust the location counter.
15 ; IF YOU PUT ALIGN BEFORE THIS DCW, IT WILL SKIP 3 BYTES, NOT JUST ONE,
16 ; TO MAKE THE ADDRESS MULTIPLE OF 4
17 four & 4,4 ; the "&" here means DCD
18 ; DCD must start at a multiple of 4 address location
19 DCD 2,1010 ; Binary positive number

```

Memory Windows:

- Memory 1
- Memory 2
- Memory 3
- Memory 4

Command Window:

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 72 Bytes (0%)
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Show or hide the Memory 1 Window

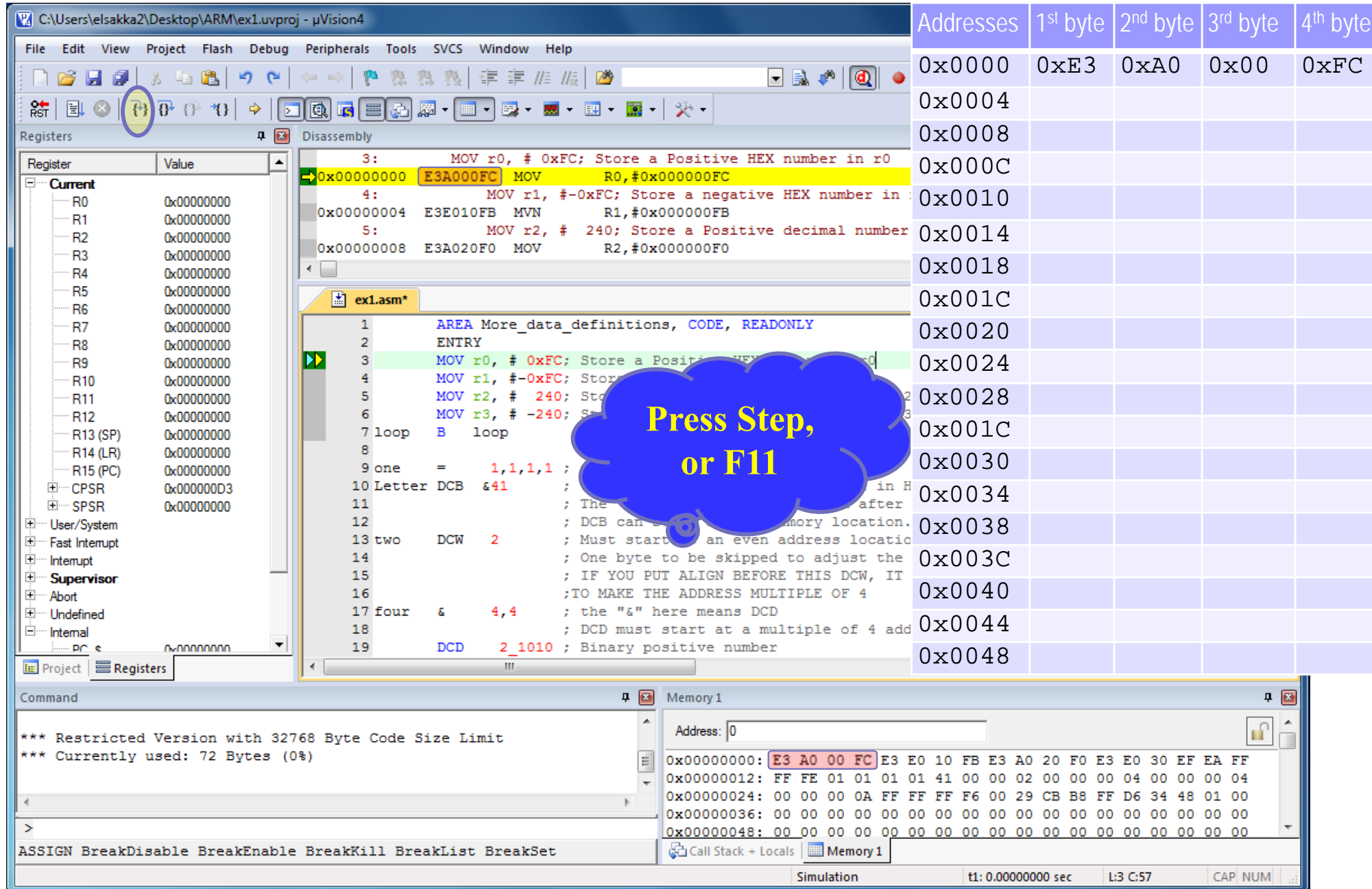
```

Call Stack + Locals:

Name	Location/Value	Type
__asm_0x0	0x00000000	void f()

Status Bar: Simulation t1: 0.00000000 sec L3 C:1 CAP NUM

Data Definition Directives



The screenshot shows the uVision4 IDE with the following components:

- Registers Window:** Lists registers R0 through R15, CPSR, and SPSR, all with values of 0x00000000.
- Disassembly Window:** Shows assembly instructions:
 - 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
- Source Window (ex1.asm):** Contains assembly code with directives like AREA, ENTRY, MOV, DCB, DCW, and DCD. A blue callout bubble points to the 'Step' button in the toolbar with the text "Press Step, or F11".
- Memory Window:** Displays a memory dump starting at address 0x00000000. The first four bytes are E3, A0, 00, FC, which correspond to the value 0xE3A000FC in hexadecimal.
- Command Window:** Shows the status: "Restricted Version with 32768 Byte Code Size Limit" and "Currently used: 72 Bytes (0%)".

Addresses	1 st byte	2 nd byte	3 rd byte	4 th byte
0x0000	0xE3	0xA0	0x00	0xFC
0x0004				
0x0008				
0x000C				
0x0010				
0x0014				
0x0018				
0x001C				
0x0020				
0x0024				
0x0028				
0x001C				
0x0030				
0x0034				
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				

Data Definition Directives

Registers

Register	Value
R0	0x000000FC
R1	0x00000000
R2	0x00000000
R3	0x00000000
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)	0x00000004
CPSR	0x000000D3
SPSR	0x00000000

Disassembly

```

3:      MOV r0, # 0xFC; Store a Positive HEX number in r0
0x00000000 E3A000FC MOV      R0,#0x000000FC
4:      MOV r1, #-0xFC; Store a negative HEX number in r1
0x00000004 E3E010FB MVN      R1,#0x000000FB
5:      MOV r2, # 240; Store a Positive decimal number in r2
0x00000008 E3A020F0 MOV      R2,#0x000000F0

```

ex1.asm*

```

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
9 one = 1,1,1,1;
10 Letter DCB &41;
11
12
13 two DCW 2;
14
15
16
17 four & 4,4; the "&" here means DCD
18
19 DCD 2_1010; Binary positive number

```

Memory1

Address	1st byte	2nd byte	3rd byte	4th byte
0x00000000	0xE3	0xA0	0x00	0xFC
0x00000004	0xE3	0xE0	0x10	0xFB
0x00000008				
0x0000000C				
0x00000010				
0x00000014				
0x00000018				
0x0000001C				
0x00000020				
0x00000024				
0x00000028				
0x0000001C				
0x00000030				
0x00000034				
0x00000038				
0x0000003C				
0x00000040				
0x00000044				
0x00000048				

Command

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 72 Bytes (0%)

```

Simulation t1: 0.00000000 sec L4 C:58 CAP NUM

Data Definition Directives

Registers

Register	Value
R0	0x000000FC
R1	0xFFFFF04
R2	0x00000000
R3	0x00000000
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)	0x00000008
CPSR	0x000000D3
SPSR	0x00000000

Disassembly

```

3:      MOV r0, # 0xFC; Store a Positive HEX number in r0
0x00000000 E3A000FC MOV     R0,#0x000000FC
4:      MOV r1, #-0xFC; Store a negative HEX number in r1
0x00000004 E3E010FB MVN     R1,#0x000000FB
5:      MOV r2, # 240; Store a Positive decimal number in r2
0x00000008 E3A020F0 MOV     R2,#0x000000F0

```

ex1.asm*

```

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
9 one = 1,1,1,1 ;
10 Letter DCB &41 ; The first character in the string is 'A'
11 ; The second character is ' ' after
12 ; DCB can store data in memory location.
13 two DCW 2 ; Must start at an even address location
14 ; One byte to be skipped to adjust the
15 ; IF YOU PUT ALIGN BEFORE THIS DCW, IT
16 ; TO MAKE THE ADDRESS MULTIPLE OF 4
17 four & 4,4 ; the "&" here means DCD
18 ; DCD must start at a multiple of 4 address
19 DCD 2,1010 ; Binary positive number

```

Memory 1

Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
0x00000000	E3	A0	00	FC	E3	E0	10	FB	E3	A0	20	F0	E3	E0	30	EF	EA	FF	FF	FE	01	01	01	01	41	00	00	02	00	04	00	00	04
0x00000012	00	00	00	0A	FF	FF	FF	F6	00	29	CB	B8	FF	D6	34	48	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0x00000024	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0x00000036	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0x00000048	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

Command

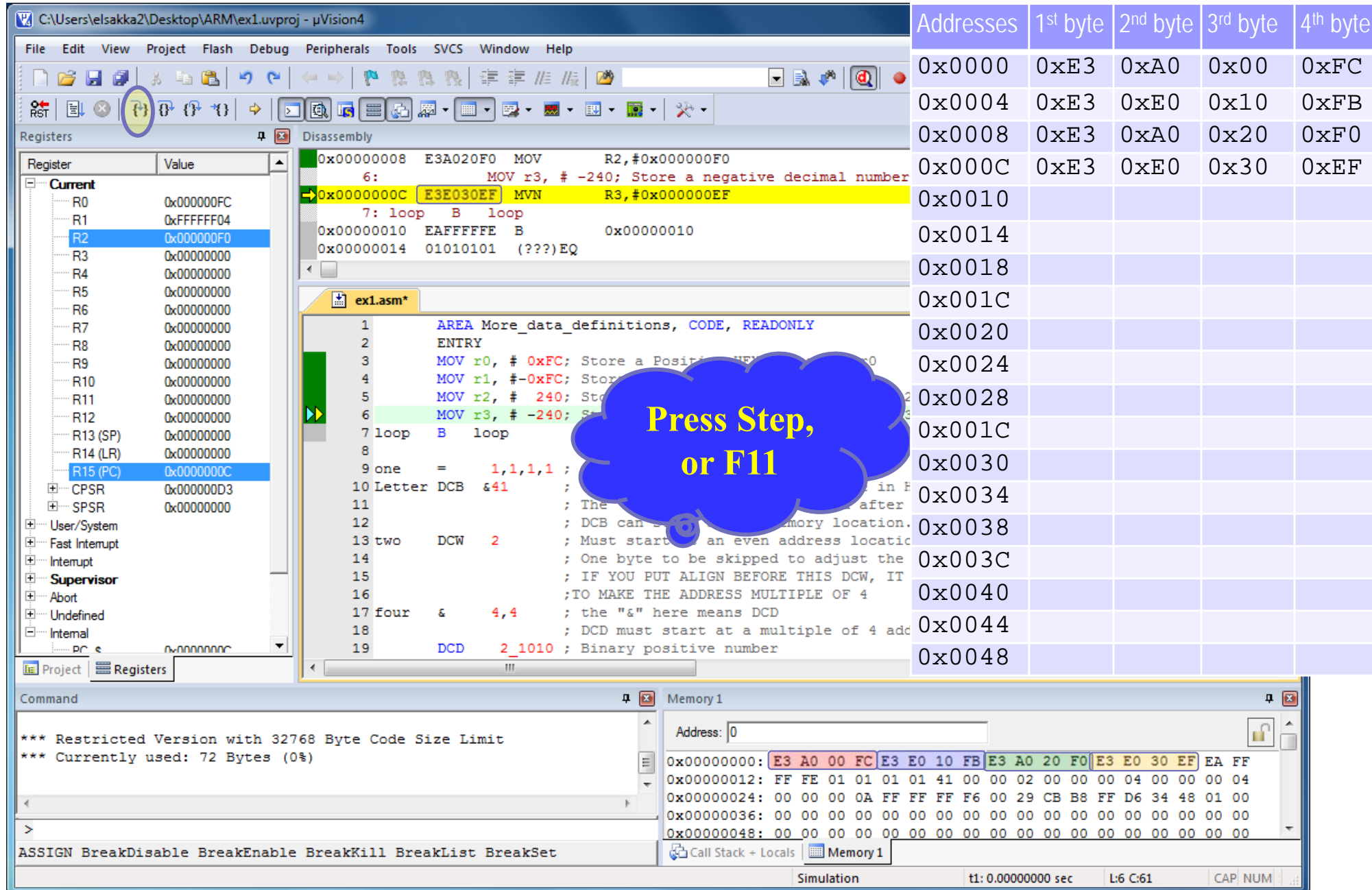
```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 72 Bytes (0%)

```

Simulation t1: 0.00000000 sec L5 C:61 CAP NUM

Data Definition Directives



The screenshot shows the uVision4 IDE with the following components:

- Registers Window:** Shows the current state of registers. R15 (PC) is highlighted with a value of 0x0000000C.
- Disassembly Window:** Shows the assembly code. The instruction at address 0x0000000C is highlighted: `MOVN R3, #0x000000EF`. A blue cloud bubble with the text "Press Step, or F11" points to this instruction.
- Assembly Window (ex1.asm):** Shows the source code. The instruction `MOV r3, #-240;` is highlighted, corresponding to the disassembly. Below it, data definitions are shown: `Letter DCB &41`, `two DCW 2`, `four & 4,4`, and `DCD 2_1010`.
- Memory Window:** Shows a memory dump starting at address 0x00000000. The dump shows the following values: `0x00000000: E3 A0 00 FC E3 E0 10 FB E3 A0 20 F0 E3 E0 30 EF EA FF`, `0x00000012: FF FE 01 01 01 01 41 00 00 02 00 00 04 00 00 04`, `0x00000024: 00 00 00 0A FF FF FF F6 00 29 CB B8 FF D6 34 48 01 00`, `0x00000036: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00`, and `0x00000048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00`.
- Command Window:** Shows the status of the simulation: `*** Restricted Version with 32768 Byte Code Size Limit` and `*** Currently used: 72 Bytes (0%)`.
- Simulation Window:** Shows the simulation status: `Simulation`, `t1: 0.00000000 sec`, `L6 C:61`, and `CAP NUM`.

Addresses	1st byte	2nd byte	3rd byte	4th byte
0x0000	0xE3	0xA0	0x00	0xFC
0x0004	0xE3	0xE0	0x10	0xFB
0x0008	0xE3	0xA0	0x20	0xF0
0x000C	0xE3	0xE0	0x30	0xEF
0x0010				
0x0014				
0x0018				
0x001C				
0x0020				
0x0024				
0x0028				
0x001C				
0x0030				
0x0034				
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				

Data Definition Directives

loop = 0x10

Press Step, or F11

Addresses	1st byte	2nd byte	3rd byte	4th byte
0x0000	0xE3	0xA0	0x00	0xFC
0x0004	0xE3	0xE0	0x10	0xFB
0x0008	0xE3	0xA0	0x20	0xF0
0x000C	0xE3	0xE0	0x30	0xEF
0x0010	0xEA	0xFF	0xFF	0xFE
0x0014				
0x0018				
0x001C				
0x0020				
0x0024				
0x0028				
0x001C				
0x0030				
0x0034				
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				

Registers

Register	Value
R0	0x000000FC
R1	0xFFFFFFFF04
R2	0x000000F0
R3	0xFFFFFFFF10
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
CPSR	0x000000D3
SPSR	0x00000000

Disassembly

```

0x00000008 E3A020F0 MOV R2,#0x000000F0
6: MOV r3, #-240; Store a negative decimal number
0x0000000C E3E030EF MVN R3,#0x000000EF
7: loop B loop
0x00000010 EAEFFFEF B 0x00000010
0x00000014 01010101 (???)EQ
  
```

ex1.asm*

```

1 AREA More_data_definitions, CODE, READONLY
2 ENTRY
3 MOV r0, # 0xFC; Store a Positive decimal number
4 MOV r1, #-0xFC; Store a negative decimal number
5 MOV r2, # 240; Store a positive decimal number
6 MOV r3, #-240; Store a negative decimal number
7 loop B loop
8
9 one = 1,1,1,1;
10 Letter DCB &41; The first character in the string is 'A'
11 ; The second character is 'a' after
12 ; DCB can be used to define memory location.
13 two DCW 2; Must start at an even address location.
14 ; One byte to be skipped to adjust the
15 ; IF YOU PUT ALIGN BEFORE THIS DCW, IT
16 ; TO MAKE THE ADDRESS MULTIPLE OF 4
17 four & 4,4; the "&" here means DCD
18 ; DCD must start at a multiple of 4 add
19 DCD 2,1010; Binary positive number
  
```

Command

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 72 Bytes (0%)
  
```

Memory 1

Address: 0

```

0x00000000: E3 A0 00 FC E3 E0 10 FB E3 A0 20 F0 E3 E0 30 EF EA FF
0x00000012: FF FE 01 01 01 01 41 00 00 02 00 00 04 00 00 04
0x00000024: 00 00 00 0A FF FF FF F6 00 29 CB B8 FF D6 34 48 01 00
0x00000036: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  
```

Simulation t1: 0.00000000 sec L7 C:16 CAP NUM

Data Definition Directives

The screenshot shows the uVision4 IDE with the following components:

- Registers:** A list of ARM registers (R0-R15, CPSR, SPSR) with their current values. R0 is 0x000000FC, R1 is 0xFFFFFFFF04, R2 is 0x000000F0, R3 is 0xFFFFFFFF10, R4 is 0x00000000, R5 is 0x00000000, R6 is 0x00000000, R7 is 0x00000000, R8 is 0x00000000, R9 is 0x00000000, R10 is 0x00000000, R11 is 0x00000000, R12 is 0x00000000, R13 (SP) is 0x00000000, R14 (LR) is 0x00000000, R15 (PC) is 0x00000010. CPSR is 0x000000D3, SPSR is 0x00000000.
- Disassembly:** Shows assembly instructions:
 - 0x00000008 E3A020F0 MOV R2,#0x000000F0
 - 6: MOV r3, #-240; Store a negative decimal number in r3
 - 0x0000000C E3E030EF MVN R3,#0x000000EF
 - 7: loop B loop
 - 0x00000010 EAEFFFE B 0x00000010
 - 0x00000014 01010101 (???)EQ
- Source Code (ex1.asm):**

```

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
9 one = 1,1,1,1 ; the "=" here means DCB
10 Letter DCB &41 ; the "&" here means an ASCII code in F
11 ; The "0x" prefix is NOT allowed after
12 ; DCB can start at any memory location.
13 two DCW 2 ; Must start at an even address location
14 ; One byte to be skipped to adjust the
15 ; IF YOU PUT ALIGN BEFORE THIS DCW, IT
16 ; TO MAKE THE ADDRESS MULTIPLE OF 4
17 four & 4,4 ; the "&" here means DCD
18 ; DCD must start at a multiple of 4 add
19 DCD 2_1010 ; Binary positive number
      
```
- Memory Dump:** A table showing memory addresses and their corresponding 4-byte values.

Addresses	1st byte	2nd byte	3rd byte	4th byte
0x0000	0xE3	0xA0	0x00	0xFC
0x0004	0xE3	0xE0	0x10	0xFB
0x0008	0xE3	0xA0	0x20	0xF0
0x000C	0xE3	0xE0	0x30	0xEF
0x0010	0xEA	0xFF	0xFF	0xFE
0x0014				
0x0018				
0x001C				
0x0020				
0x0024				
0x0028				
0x001C				
0x0030				
0x0034				
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				
- Command Window:**

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 72 Bytes (0%)
      
```
- Memory Window:** Shows a memory dump starting at address 0. The first few lines are:


```

0x00000000: E3 A0 00 FC E3 E0 10 FB E3 A0 20 F0 E3 E0 30 EF EA FF
0x00000012: FF FE 01 01 01 01 41 00 00 02 00 00 04 00 00 04
0x00000024: 00 00 00 0A FF FF FF F6 00 29 CB B8 FF D6 34 48 01 00
0x00000036: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      
```


Data Definition Directives

The "=" here means DCB

loop = 0x10

one = 0x14

Addresses	1st byte	2nd byte	3rd byte	4th byte
0x0000	0xE3	0xA0	0x00	0xFC
0x0004	0xE3	0xE0	0x10	0xFB
0x0008	0xE3	0xA0	0x20	0xF0
0x000C	0xE3	0xE0	0x30	0xEF
0x0010	0xEA	0xFF	0xFF	0xFE
0x0014	0x01	0x01	0x01	0x01
0x0018				
0x001C				
0x0020				
0x0024				
0x0028				
0x001C				
0x0030				
0x0034				
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				

```

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
9  one = 1,1,1,1; the "=" here means DCB
10 Letter DCB &41; the "&" here means an ASCII code in F
11; The "0x" prefix is NOT allowed after
12; DCB can start at any memory location.
13 two DCW 2; Must start at an even address locatio
14; One byte to be skipped to adjust the
15; IF YOU PUT ALIGN BEFORE THIS DCW, IT
16; TO MAKE THE ADDRESS MULTIPLE OF 4
17 four & 4,4; the "&" here means DCD
18; DCD must start at a multiple of 4 add
19 DCD 2_1010; Binary positive number
  
```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 72 Bytes (0%)

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet

Memory 1

Address: 0

```

0x00000000: E3 A0 00 FC E3 E0 10 FB E3 A0 20 F0 E3 E0 30 EF EA FF
0x00000012: FF FE 01 01 01 01 41 00 00 02 00 00 04 00 00 04
0x00000024: 00 00 00 0A FF FF FF F6 00 29 CB B8 FF D6 34 48 01 00
0x00000036: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  
```

Simulation t1: 0.00000000 sec L9 C:46 CAP NUM

Data Definition Directives

The “&” means a HEX number

Annotations in the image:

- loop = 0x10** (points to line 7: loop B loop)
- one = 0x14** (points to line 9: one = 1,1,1,1 ; the "=" here means DCB)
- Letter = 0x18** (points to line 10: Letter DCB &41 ; the "&" here means an ASCII code in H)

Addresses	1 st byte	2 nd byte	3 rd byte	4 th byte
0x0000	0xE3	0xA0	0x00	0xFC
0x0004	0xE3	0xE0	0x10	0xFB
0x0008	0xE3	0xA0	0x20	0xF0
0x000C	0xE3	0xE0	0x30	0xEF
0x0010	0xEA	0xFF	0xFF	0xFE
0x0014	0x01	0x01	0x01	0x01
0x0018	0x41			
0x001C				
0x0020				
0x0024				
0x0028				
0x001C				
0x0030				
0x0034				
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				

```

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
9 one = 1,1,1,1 ; the "=" here means DCB
10 Letter DCB &41 ; the "&" here means an ASCII code in H
11 ; The "0x" prefix is NOT allowed after
12 ; DCB can start at any memory location.
13 two DCW 2 ; Must start at an even address location
14 ; One byte to be skipped to adjust the
15 ; IF YOU PUT ALIGN BEFORE THIS DCW, IT
16 ; TO MAKE THE ADDRESS MULTIPLE OF 4
17 four & 4,4 ; the "&" here means DCD
18 ; DCD must start at a multiple of 4 add
19 DCD 2_1010 ; Binary positive number
  
```

Memory 1

Address: 0

```

0x00000000: E3 A0 00 FC E3 E0 10 FB E3 A0 20 F0 E3 E0 30 EF EA FF
0x00000012: FF FE 01 01 01 01 41 00 00 02 00 00 00 04 00 00 04
0x00000024: 00 00 00 0A FF FF FF F6 00 29 CB B8 FF D6 34 48 01 00
0x00000036: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  
```

Simulation t1: 0.00000000 sec L9 C:46 CAP NUM

Data Definition Directives

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:** Shows the current state of ARM registers (R0-R15, CPSR, SPSR).
- Disassembly Window:** Shows the assembly code being executed, with a yellow highlight on the instruction at address 0x00000014: `01010101 (???)EQ`.
- Source Code Window (ex1.asm):** Contains the assembly source code with several data definition directives:
 - `loop = 0x10` (Annotation)
 - `one = 0x14` (Annotation)
 - `Letter = 0x18` (Annotation)
 - `two = 0x1A` (Annotation)
- Memory Window:** Displays the memory dump starting at address 0x00000000. The dump shows the following values:

Addresses	1st byte	2nd byte	3rd byte	4th byte
0x0000	0xE3	0xA0	0x00	0xFC
0x0004	0xE3	0xE0	0x10	0xFB
0x0008	0xE3	0xA0	0x20	0xF0
0x000C	0xE3	0xE0	0x30	0xEF
0x0010	0xEA	0xFF	0xFF	0xFE
0x0014	0x01	0x01	0x01	0x01
0x0018	0x41	0x00	0x00	0x02
0x001C				
0x0020				
0x0024				
0x0028				
0x001C				
0x0030				
0x0034				
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				

Annotations:

- Must start at an even address location:** A red cloud-shaped annotation pointing to the memory dump.
- Annotations pointing to memory dump:**
 - `loop = 0x10` points to the value 0x10 at address 0x0004.
 - `one = 0x14` points to the value 0x14 at address 0x0008.
 - `Letter = 0x18` points to the value 0x18 at address 0x000C.
 - `two = 0x1A` points to the value 0x1A at address 0x0010.

Source Code (ex1.asm):

```

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
9  one = 1,1,1,1 ; the "=" here means DCB
10 Letter DCB &41 ; the "&" here means an ASCII code in H
11 ; The "0x" prefix is NOT allowed after
12 ; DCB can start at any memory location.
13 two DCW 2 ; Must start at an even address locatio
14 ; One byte to be skipped to adjust the
15 ; IF YOU PUT ALIGN BEFORE THIS DCW, IT
16 ; TO MAKE THE ADDRESS MULTIPLE OF 4
17 four & 4,4 ; the "&" here means DCD
18 ; DCD must start at a multiple of 4 add
19 DCD 2_1010 ; Binary positive number

```

Command Window:

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 72 Bytes (0%)

```

Memory Window:

```

Address: 0
0x00000000: E3 A0 00 FC E3 E0 10 FB E3 A0 20 F0 E3 E0 30 EF EA FF
0x00000012: FF FE 01 01 01 01 41 00 00 02 00 00 04 00 00 04
0x00000024: 00 00 00 0A FF FF FF F6 00 29 CB B8 FF D6 34 48 01 00
0x00000036: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

Simulation Status: Simulation, t1: 0.00000000 sec, L9 C:46, CAP NUM

Data Definition Directives

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:** Shows the current state of ARM registers R0 through R15, CPSR, and SPSR.
- Disassembly Window:** Shows the assembly code being executed, including instructions like `EAFFFFE B`, `01010101 (???)EQ`, `41000002 (???)MI`, and `ANDEQ R0,R0,R4`.
- Source Code Window (ex.asm*):** Contains the assembly source code with data definitions:


```

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
9 one = 1,1,1,1 ; the "=" here means DCB
10 Letter DCB &41 ; the "&" here means an ASCII code in H
11 ; The "0x" prefix is NOT allowed after
12 ; DCB can start at any memory location.
13 two DCW 2 ; Must start at an even address location
14 ; One byte to be skipped to adjust the
15 ; IF YOU PUT ALIGN BEFORE THIS DCW, IT
16 ; TO MAKE THE ADDRESS MULTIPLE OF 4
17 four & 4,4 ; the "&" here means DCD
18 ; DCD must start at a multiple of 4 add
19 DCD 2_1010 ; Binary positive number
      
```
- Memory Window:** Shows the memory dump starting at address 0x00000000, displaying the hexadecimal values of the data defined in the source code.

Annotations:

- loop = 0x10:** Points to the `01010101` value in the disassembly window.
- one = 0x14:** Points to the `41000002` value in the disassembly window.
- Letter = 0x18:** Points to the `&41` value in the source code window.
- two = 0x1A:** Points to the `2` value in the source code window.
- four = 0x1C:** Points to the `& 4,4` value in the source code window.
- The "&" here means DCD:** A callout pointing to the `&` in line 17 of the source code.

Memory Dump (Address 0x00000000):

Address	1st byte	2nd byte	3rd byte	4th byte
0x0000	0xE3	0xA0	0x00	0xFC
0x0004	0xE3	0xE0	0x10	0xFB
0x0008	0xE3	0xA0	0x20	0xF0
0x000C	0xE3	0xE0	0x30	0xEF
0x0010	0xEA	0xFF	0xFF	0xFE
0x0014	0x01	0x01	0x01	0x01
0x0018	0x41	0x00	0x00	0x02
0x001C	0x00	0x00	0x00	0x04
0x0020	0x00	0x00	0x00	0x04
0x0024				
0x0028				
0x001C				
0x0030				
0x0034				
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				

Data Definition Directives

The screenshot displays the uVision4 IDE interface. The left pane shows the Register window with R0-R15 values. The main pane shows the Disassembly window with assembly instructions. The right pane shows the Memory window with a hex dump. Annotations highlight specific data definition directives and their values.

Annotations:

- loop = 0x10**: Points to the value 0x10 in the memory dump at address 0x0004.
- one = 0x14**: Points to the value 0x14 in the memory dump at address 0x0008.
- Letter = 0x18**: Points to the value 0x18 in the memory dump at address 0x000C.
- two = 0x1A**: Points to the value 0x1A in the memory dump at address 0x0018.
- four = 0x1C**: Points to the value 0x1C in the memory dump at address 0x001C.

Assembly Code (ex1.asm):

```

12      ; DCB can start at any memory location.
13 two   DCW  2      ; Must start at an even address location.
14      ; One byte to be skipped to align the
15      ; IF YOU PUT ALIGN BEFORE THIS DCW, IT
16      ; TO MAKE THE ALIGN OF 4
17 four  & 4,4      ; the "&" here means DCD
18      ; DCD must start at a multiple of 4 address
19      DCD  2_1010 ; Binary positive number
20      DCD -2_1010 ; Binary negative number
21      DCD  8_12345670 ; Octal positive number
22      DCD -8_12345670 ; Octal negative number
23
24      DCB  1      ; Any data directive can be without label
25 data_1 SPACE 5    ; reserves a ZEROED 5 bytes block of memory
26 data_2 % 5        ; the "%" here means SPACE
27      ALIGN      ; ADVANCE THE LOCATION COUNTER TO THE NEXT
28      ; OF 4 ADDRESS LOCATION
29 data_3 SPACE 5
30      END
  
```

Memory Dump (Memory 1):

Address	1st byte	2nd byte	3rd byte	4th byte
0x0000	0xE3	0xA0	0x00	0xFC
0x0004	0xE3	0xE0	0x10	0xFB
0x0008	0xE3	0xA0	0x20	0xF0
0x000C	0xE3	0xE0	0x30	0xEF
0x0010	0xEA	0xFF	0xFF	0xFE
0x0014	0x01	0x01	0x01	0x01
0x0018	0x41	0x00	0x00	0x02
0x001C	0x00	0x00	0x00	0x04
0x0020	0x00	0x00	0x00	0x04
0x0024				
0x0028				
0x001C				
0x0030				
0x0034				
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				

Command Window:

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 72 Bytes (0%)
  
```

Simulation Status: Simulation, t1: 0.00000000 sec, L9 C:46, CAP NUM

Data Definition Directives

The screenshot displays the uVision4 IDE interface. The left pane shows the Register window with values for R0 through R15, CPSR, and SPSR. The main pane shows the Disassembly window with assembly code. The right pane shows the Memory window with a hex dump of memory addresses from 0x0000 to 0x0048.

Annotations:

- loop = 0x10**: Points to the value 0x10 in the memory dump at address 0x0004.
- one = 0x14**: Points to the value 0x14 in the memory dump at address 0x0008.
- Letter = 0x18**: Points to the value 0x18 in the memory dump at address 0x000C.
- two = 0x1A**: Points to the value 0x1A in the memory dump at address 0x001C.
- four = 0x1C**: Points to the value 0x1C in the memory dump at address 0x0028.

Assembly Code (ex1.asm):

```

12      ; DCB can start at any memory location.
13 two   DCW   2      ; Must start at an even address location
14      ; One byte to be skipped to align the
15      ; IF YOU PUT ALIGN BEFORE THIS DCW, IT
16      ; TO MAKE THE ALIGN OF 4
17 four   &      4,4   ; the "&" here means DCD
18      ; DCD must start at a multiple of 4 address
19      DCD    2_1010 ; Binary positive number
20      DCD    -2_1010 ; Binary negative number
21      DCD    8_12345670 ; Octal positive number
22      DCD    -8_12345670 ; Octal negative number
23
24      DCB    1      ; Any data directive can be without label
25 data_1 SPACE 5      ; reserves a ZEROED 5 bytes block of memory
26 data_2 %      5      ; the "%" here means SPACE
27      ALIGN      ; ADVANCE THE LOCATION COUNTER TO THE NEXT
28      ; OF 4 ADDRESS LOCATION
29 data_3 SPACE 5
30      END
  
```

Memory Dump:

Addresses	1st byte	2nd byte	3rd byte	4th byte
0x0000	0xE3	0xA0	0x00	0xFC
0x0004	0xE3	0xE0	0x10	0xFB
0x0008	0xE3	0xA0	0x20	0xF0
0x000C	0xE3	0xE0	0x30	0xEF
0x0010	0xEA	0xFF	0xFF	0xFE
0x0014	0x01	0x01	0x01	0x01
0x0018	0x41	0x00	0x00	0x02
0x001C	0x00	0x00	0x00	0x04
0x0020	0x00	0x00	0x00	0x04
0x0024	0x00	0x00	0x00	0x0A
0x0028				
0x001C				
0x0030				
0x0034				
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				

Command Window:

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 72 Bytes (0%)
  
```

Memory Window:

Address: 0

0x00000000: E3 A0 00 FC E3 E0 10 FB E3 A0 20 F0 E3 E0 30 EF EA FF

0x00000012: FF FE 01 01 01 01 41 00 00 02 00 00 00 04 00 00 04

0x00000024: 00 00 00 0A FF FF FF F6 00 29 CB B8 FF D6 34 48 01 00

0x00000036: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x00000048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Data Definition Directives

The screenshot displays the uVision4 IDE interface. The left pane shows the Register window with values for R0 through R15, CPSR, and SPSR. The main pane shows the Disassembly window with assembly code. The right pane shows a memory dump table.

Annotations:

- loop = 0x10**: Points to the value 0x10 in the memory dump at address 0x0008.
- one = 0x14**: Points to the value 0x14 in the memory dump at address 0x000C.
- Letter = 0x18**: Points to the value 0x18 in the memory dump at address 0x0010.
- two = 0x1A**: Points to the value 0x1A in the memory dump at address 0x001C.
- four = 0x1C**: Points to the value 0x1C in the memory dump at address 0x0020.

Assembly Code (ex1.asm):

```

12      ; DCB can start at any memory location.
13 two   DCW   2      ; Must start at an even address location
14      ; One byte to be skipped to align the
15      ; IF YOU PUT ALIGN BEFORE THIS DCW, IT
16      ; TO MAKE THE ALIGN OF 4
17 four   &      4,4   ; the "&" here means DCB
18      ; DCB must start at a multiple of 4 address
19      DCD   2 1010   ; Binary positive number
20      DCD  -2 1010   ; Binary negative number
21      DCD   8 12345670 ; Octal positive number
22      DCD  -8 12345670 ; Octal negative number
23
24      DCB   1      ; Any data directive can be without label
25 data_1 SPACE 5     ; reserves a ZEROED 5 bytes block of memory
26 data_2 %      5     ; the "%" here means SPACE
27      ALIGN      ; ADVANCE THE LOCATION COUNTER TO THE NEXT
28      ; OF 4 ADDRESS LOCATION
29 data_3 SPACE 5
30      END

```

Memory Dump Table:

Addresses	1st byte	2nd byte	3rd byte	4th byte
0x0000	0xE3	0xA0	0x00	0xFC
0x0004	0xE3	0xE0	0x10	0xFB
0x0008	0xE3	0xA0	0x20	0xF0
0x000C	0xE3	0xE0	0x30	0xEF
0x0010	0xEA	0xFF	0xFF	0xFE
0x0014	0x01	0x01	0x01	0x01
0x0018	0x41	0x00	0x00	0x02
0x001C	0x00	0x00	0x00	0x04
0x0020	0x00	0x00	0x00	0x04
0x0024	0x00	0x00	0x00	0x0A
0x0028	0xFF	0xFF	0xFF	0xF6
0x001C				
0x0030				
0x0034				
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				

Command Window:

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 72 Bytes (0%)

```

Memory Window:

Address: 0

0x00000000:	E3 A0 00 FC	E3 E0 10 FB	E3 A0 20 F0	E3 E0 30 EF	EA FF
0x00000012:	FF FE	01 01 01 01	41 00	00 02	00 00 04 00 00 00 04
0x00000024:	00 00 00 0A	FF FF FF F6	00 29 CB B8	FF D6 34 48	01 00
0x00000036:	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
0x00000048:	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00

Simulation t1: 0.00000000 sec L9 C:46 CAP NUM

Data Definition Directives

The screenshot displays the uVision4 IDE interface. The left pane shows the Register window with R0-R15 and CPSR/SPSR. The main pane shows the Disassembly window with assembly code. The right pane shows a memory dump table.

Annotations:

- loop = 0x10**: Points to the value 0x10 in the memory dump at address 0x0008.
- one = 0x14**: Points to the value 0x14 in the memory dump at address 0x000C.
- Letter = 0x18**: Points to the value 0x18 in the memory dump at address 0x0010.
- two = 0x1A**: Points to the value 0x1A in the memory dump at address 0x001C.
- four = 0x1C**: Points to the value 0x1C in the memory dump at address 0x0020.

Assembly Code (ex1.asm):

```

12      ; DCB can start at any memory location.
13 two   DCW   2      ; Must start at an even address location
14      ; One byte to be skipped to align the
15      ; IF YOU PUT ALIGN BEFORE THIS DCW, IT
16      ; TO MAKE THE ADDRESS OF 4
17 four   &      4,4   ; the "&" here means DCB
18      ; DCB must start at a multiple of 4 address
19      DCD   2_1010 ; Binary positive number
20      DCD  -2_1010 ; Binary negative number
21      DCD   8_12345670 ; Octal positive number
22      DCD  -8_12345670 ; Octal negative number
23
24      DCB   1      ; Any data directive can be without label
25 data_1 SPACE 5    ; reserves a ZEROED 5 bytes block of memory
26 data_2 %      5    ; the "%" here means SPACE
27      ALIGN      ; ADVANCE THE LOCATION COUNTER TO THE NEXT
28      ; OF 4 ADDRESS LOCATION
29 data_3 SPACE 5
30      END
  
```

Memory Dump Table:

Addresses	1 st byte	2 nd byte	3 rd byte	4 th byte
0x0000	0xE3	0xA0	0x00	0xFC
0x0004	0xE3	0xE0	0x10	0xFB
0x0008	0xE3	0xA0	0x20	0xF0
0x000C	0xE3	0xE0	0x30	0xEF
0x0010	0xEA	0xFF	0xFF	0xFE
0x0014	0x01	0x01	0x01	0x01
0x0018	0x41	0x00	0x00	0x02
0x001C	0x00	0x00	0x00	0x04
0x0020	0x00	0x00	0x00	0x04
0x0024	0x00	0x00	0x00	0x0A
0x0028	0xFF	0xFF	0xFF	0xF6
0x001C	0x00	0x29	0xCB	0xB8
0x0030				
0x0034				
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				

Command Window:

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 72 Bytes (0%)
  
```

Memory Window:

Address: 0

```

0x00000000: E3 A0 00 FC E3 E0 10 FB E3 A0 20 F0 E3 E0 30 EF EA FF
0x00000012: FF FE 01 01 01 01 41 00 00 02 00 00 00 04 00 00 04
0x00000024: 00 00 00 0A FF FF FF F6 00 29 CB B8 FF D6 34 48 01 00
0x00000036: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  
```


Data Definition Directives

The screenshot shows the uVision4 IDE with the following components:

- Registers Panel:** Shows the current state of ARM registers. R0 is 0x000000FC, R1 is 0xFFFFF04, R2 is 0x000000F0, R3 is 0xFFFFF10, R4 is 0x00000000, R5 is 0x00000000, R6 is 0x00000000, R7 is 0x00000000, R8 is 0x00000000, R9 is 0x00000000, R10 is 0x00000000, R11 is 0x00000000, R12 is 0x00000000, R13 (SP) is 0x00000000, R14 (LR) is 0x00000000, R15 (PC) is 0x00000010. CPSR is 0x000000D3 and SPSR is 0x00000000.
- Disassembly Panel:** Shows assembly instructions. A yellow highlight is on the instruction at address 0x00000040: `ANDEQ R0,R0,R0`.
- Source File Panel (ex.asm):**

```

12      ; DCB can start at any memory location.
13 two   DCW   2      ; Must start at an even address location.
14      ; One byte to be skipped request the
15      ; IF YOU PUT ALIGN BEFORE THIS DCW, IT
16      ; TO MAKE THE ALIGNMENT OF 4
17 four   &      4,4    ; the "&" here means DCB
18      ; DCB must start at a multiple of 4 add
19      DCD   2_1010    ; Binary positive number
20      DCD  -2_1010    ; Binary negative number
21      DCD   8_12345670 ; Octal positive number
22      DCD  -8_12345670 ; Octal negative number
23
24      DCB   1      ; Any data directive can be without label
25 data_1 SPACE 5    ; reserves a ZEROED 5 bytes block of memory
26 data_2 %   5      ; the "%" here means SPACE
27      ALIGN      ; ADVANCE THE LOCATION COUNTER TO THE NEXT
28      ; OF 4 ADDRESS LOCATION
29 data_3 SPACE 5
30      END

```
- Memory Panel:** Shows a memory dump starting at address 0x00000000. The first row of data is: `E3 A0 00 FC E3 E0 10 FB E3 A0 20 F0 E3 E0 30 EF EA FF`.
- Annotations:**
 - loop = 0x10:** Points to the value 0x10 in the memory dump at address 0x00000004.
 - one = 0x14:** Points to the value 0x14 in the memory dump at address 0x00000008.
 - Letter = 0x18:** Points to the value 0x18 in the memory dump at address 0x0000000C.
 - two = 0x1A:** Points to the value 0x1A in the memory dump at address 0x00000010.
 - four = 0x1C:** Points to the value 0x1C in the memory dump at address 0x00000014.

Data Definition Directives

The screenshot displays the uVision4 IDE interface. The left pane shows the Register window with R0-R15 and CPSR/SPSR. The main pane shows the Disassembly window with assembly code. The right pane shows a memory dump table. Annotations with green callouts point to specific values in the code and memory dump.

Annotations:

- loop = 0x10**: Points to the value 0x10 in the memory dump at address 0x0004.
- one = 0x14**: Points to the value 0x14 in the memory dump at address 0x0008.
- Letter = 0x18**: Points to the value 0x18 in the memory dump at address 0x000C.
- two = 0x1A**: Points to the value 0x1A in the memory dump at address 0x0014.
- four = 0x1C**: Points to the value 0x1C in the memory dump at address 0x0018.

Assembly Code (ex.asm):

```

12      ; DCB can start at any memory location.
13 two   DCW   2      ; Must start at an even address location.
14      ; One byte to be skipped request the
15      ; IF YOU PUT ALIGN BEFORE THIS DCW, IT
16      ; TO MAKE THE ALIGNMENT OF 4
17 four   &      4,4   ; the "&" here means DCB
18      ; DCB must start at a multiple of 4 address
19      DCD   2_1010   ; Binary positive number
20      DCD  -2_1010   ; Binary negative number
21      DCD   8_12345670 ; Octal positive number
22      DCD  -8_12345670 ; Octal negative number
23
24      DCB   1      ; Any data directive can be without label
25 data_1 SPACE 5   ; reserves a ZEROED 5 bytes block of memory
26 data_2 %      5   ; the "%" here means SPACE
27      ALIGN      ; ADVANCE THE LOCATION COUNTER TO THE NEXT
28      ; OF 4 ADDRESS LOCATION
29 data_3 SPACE 5
30      END
  
```

Memory Dump Table:

Addresses	1 st byte	2 nd byte	3 rd byte	4 th byte
0x0000	0xE3	0xA0	0x00	0xFC
0x0004	0xE3	0xE0	0x10	0xFB
0x0008	0xE3	0xA0	0x20	0xF0
0x000C	0xE3	0xE0	0x30	0xEF
0x0010	0xEA	0xFF	0xFF	0xFE
0x0014	0x01	0x01	0x01	0x01
0x0018	0x41	0x00	0x00	0x02
0x001C	0x00	0x00	0x00	0x04
0x0020	0x00	0x00	0x00	0x04
0x0024	0x00	0x00	0x00	0x0A
0x0028	0xFF	0xFF	0xFF	0xF6
0x001C	0x00	0x29	0xCB	0xB8
0x0030	0xFF	0xD6	0x34	0x48
0x0034	0x01			
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				

Command Window:

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 72 Bytes (0%)
  
```

Memory Window:

Address: 0

```

0x00000000: E3 A0 00 FC E3 E0 10 FB E3 A0 20 F0 E3 E0 30 EF EA FF
0x00000012: FF FE 01 01 01 01 41 00 00 02 00 00 04 00 00 04
0x00000024: 00 00 00 0A FF FF FF F6 00 29 CB B8 FF D6 34 48 01 00
0x00000036: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  
```

Simulation | t1: 0.00000000 sec | L9 C:46 | CAP NUM

Data Definition Directives

The screenshot displays the uVision4 IDE interface. The left pane shows the Register window with various registers and their values. The main pane shows the Disassembly window with assembly code. The right pane shows the Memory window with a memory dump. Annotations in green boxes point to specific values in the code and memory dump.

Annotations:

- loop = 0x10**: Points to the value 0x10 in the memory dump at address 0x0004.
- one = 0x14**: Points to the value 0x14 in the memory dump at address 0x0008.
- Letter = 0x18**: Points to the value 0x18 in the memory dump at address 0x000C.
- two = 0x1A**: Points to the value 0x1A in the memory dump at address 0x001C.
- four = 0x1C**: Points to the value 0x1C in the memory dump at address 0x0028.
- data_1 = 0x35**: Points to the value 0x35 in the memory dump at address 0x0034.

Assembly Code (ex.asm):

```

12      ; DCB can start at any memory location.
13 two   DCW   2      ; Must start at an even address location.
14      ; One byte to be skipped request the
15      ; IF YOU PUT ALIGN BEFORE THIS DCW, IT
16      ; TO MAKE THE ADDRESS OF 4
17 four   &      4,4   ; the "&" here means DCB
18      ; DCB must start at a multiple of 4 add
19      DCD   2_1010 ; Binary positive number
20      DCD   -2_1010 ; Binary negative number
21      DCD   8_12345670 ; Octal positive number
22      DCD   -8_12345670 ; Octal negative number
23
24      DCB   1      ; Any data directive can be without label
25 data_1 SPACE 5    ; reserves a ZEROED 5 bytes block of memory
26 data_2 %      5    ; the "%" here means SPACE
27      ALIGN      ; ADVANCE THE LOCATION COUNTER TO THE NEXT
28      ; OF 4 ADDRESS LOCATION
29 data_3 SPACE 5
30      END
  
```

Memory Dump (Memory 1):

Address	1st byte	2nd byte	3rd byte	4th byte
0x0000	0xE3	0xA0	0x00	0xFC
0x0004	0xE3	0xE0	0x10	0xFB
0x0008	0xE3	0xA0	0x20	0xF0
0x000C	0xE3	0xE0	0x30	0xEF
0x0010	0xEA	0xFF	0xFF	0xFE
0x0014	0x01	0x01	0x01	0x01
0x0018	0x41	0x00	0x00	0x02
0x001C	0x00	0x00	0x00	0x04
0x0020	0x00	0x00	0x00	0x04
0x0024	0x00	0x00	0x00	0x0A
0x0028	0xFF	0xFF	0xFF	0xF6
0x001C	0x00	0x29	0xCB	0xB8
0x0030	0xFF	0xD6	0x34	0x48
0x0034	0x01	0x00	0x00	0x00
0x0038	0x00	0x00		
0x003C				
0x0040				
0x0044				
0x0048				

Command Window:

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 72 Bytes (0%)
  
```

Simulation Status:

Simulation t1: 0.00000000 sec L9 C:46 CAP NUM

Data Definition Directives

The screenshot displays the uVision4 IDE interface. The left pane shows the Register window with R0-R15 and CPSR/SPSR. The main pane shows the Disassembly window with assembly code. The bottom pane shows the Memory window with a hex dump.

Annotations:

- loop = 0x10**: Points to the value 0x10 in the memory dump at address 0x00000004.
- one = 0x14**: Points to the value 0x14 in the memory dump at address 0x00000008.
- Letter = 0x18**: Points to the value 0x18 in the memory dump at address 0x0000000C.
- two = 0x1A**: Points to the value 0x1A in the memory dump at address 0x00000014.
- four = 0x1C**: Points to the value 0x1C in the memory dump at address 0x00000018.
- data_1 = 0x35**: Points to the value 0x35 in the memory dump at address 0x00000034.
- data_2 = 0x3A**: Points to the value 0x3A in the memory dump at address 0x00000038.

Assembly Code (ex1.asm):

```

12      ; DCB can start at any memory location.
13 two   DCW   2      ; Must start at an even address location.
14      ; One byte to be skipped request the
15      ; IF YOU PUT ALIGN BEFORE THIS DCW, IT
16      ; TO MAKE THE ADDRESS IS A MULTIPLE OF 4
17 four   &      4,4   ; the "&" here means SPACE
18      ; DCB must start at a multiple of 4 address
19      DCD   2_1010 ; Binary positive number
20      DCD   -2_1010 ; Binary negative number
21      DCD   8_12345670 ; Octal positive number
22      DCD   -8_12345670 ; Octal negative number
23
24      DCB   1      ; Any data definition can be without label
25 data_1 SPACE 5    ; reserves a ZEROED 5 bytes block of memory
26 data_2 % 5        ; the "%" here means SPACE
27      ALIGN      ; ADVANCE THE LOCATION COUNTER TO THE NEXT
28      ; OF 4 ADDRESS LOCATION
29 data_3 SPACE 5
30      END
  
```

Memory Dump (Memory 1):

Address	1st byte	2nd byte	3rd byte	4th byte
0x00000000	0xE3	0xA0	0x00	0xFC
0x00000004	0xE3	0xE0	0x10	0xFB
0x00000008	0xE3	0xA0	0x20	0xF0
0x0000000C	0xE3	0xE0	0x30	0xEF
0x00000010	0xEA	0xFF	0xFF	0xFE
0x00000014	0x01	0x01	0x01	0x01
0x00000018	0x41	0x00	0x00	0x02
0x0000001C	0x00	0x00	0x00	0x04
0x00000020	0x00	0x00	0x00	0x04
0x00000024	0x00	0x00	0x00	0x0A
0x00000028	0xFF	0xFF	0xFF	0xF6
0x0000001C	0x00	0x29	0xCB	0xB8
0x00000030	0xFF	0xD6	0x34	0x48
0x00000034	0x01	0x00	0x00	0x00
0x00000038	0x00	0x00	0x00	0x00
0x0000003C	0x00	0x00	0x00	0x00
0x00000040				
0x00000044				
0x00000048				

Memory Window (Memory 1):

Address: 0

0x00000000: E3 A0 00 FC E3 E0 10 FB E3 A0 20 F0 E3 E0 30 EF EA FF

0x00000012: FF FE 01 01 01 01 41 00 00 02 00 00 04 00 00 04

0x00000024: 00 00 00 0A FF FF FF F6 00 29 CB B8 FF D6 34 48 01 00

0x00000036: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x00000048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Simulation | t1: 0.00000000 sec | L9 C:46 | CAP NUM

The "%" here means SPACE

Data Definition Directives

The screenshot shows the uVision4 IDE with the following components:

- Registers Panel:** Shows the current state of ARM registers. R0 is 0x000000FC, R1 is 0xFFFFF04, R2 is 0x000000F0, R3 is 0xFFFFF10, R4 is 0x00000000, R5 is 0x00000000, R6 is 0x00000000, R7 is 0x00000000, R8 is 0x00000000, R9 is 0x00000000, R10 is 0x00000000, R11 is 0x00000000, R12 is 0x00000000, R13 (SP) is 0x00000000, R14 (LR) is 0x00000000, R15 (PC) is 0x00000010, CPSR is 0x000000D3, and SPSR is 0x00000000.
- Disassembly Panel:** Shows the assembly code being executed. The current instruction is `ANDEQ R0,R0,R0` at address 0x00000040.
- Source File Panel:** Shows the assembly code file `ex1.asm`. The code includes directives for defining data: `DCW`, `DCD`, and `DCB`. Annotations highlight specific values: `loop = 0x10`, `one = 0x14`, `Letter = 0x18`, `two = 0x1A`, `four = 0x1C`, `data_1 = 0x35`, and `data_2 = 0x3A`.
- Memory Panel:** Shows the memory dump starting at address 0x00000000. The dump is organized into rows of 16 bytes each. Annotations highlight specific values: `0x00000000: E3 A0 00 FC`, `0x00000004: E3 E0 10 FB`, `0x00000008: E3 A0 20 F0`, `0x0000000C: E3 E0 30 EF`, `0x00000010: EA FF`, `0x00000014: FF FE 01 01 01 01 41 00`, `0x00000018: 00 00 00 02 00 00 00 04`, `0x0000001C: 00 00 00 00 0A FF FF FF F6`, `0x00000020: 00 29 CB B8 FF D6 34 48`, `0x00000024: 01 00`, `0x00000028: 00 00 00 00 00 00 00 00`, `0x0000002C: 00 00 00 00 00 00 00 00`, `0x00000030: 00 00 00 00 00 00 00 00`, `0x00000034: 00 00 00 00 00 00 00 00`, `0x00000038: 00 00 00 00 00 00 00 00`, `0x0000003C: 00 00 00 00 00 00 00 00`, `0x00000040: 00 00 00 00 00 00 00 00`, `0x00000044: 00 00 00 00 00 00 00 00`, `0x00000048: 00 00 00 00 00 00 00 00`.
- Annotations:**
 - `loop = 0x10` points to the value 0x10 in the memory dump at address 0x00000008.
 - `one = 0x14` points to the value 0x14 in the memory dump at address 0x00000010.
 - `Letter = 0x18` points to the value 0x18 in the memory dump at address 0x00000012.
 - `two = 0x1A` points to the value 0x1A in the memory dump at address 0x00000014.
 - `four = 0x1C` points to the value 0x1C in the memory dump at address 0x00000016.
 - `data_1 = 0x35` points to the value 0x35 in the memory dump at address 0x00000018.
 - `data_2 = 0x3A` points to the value 0x3A in the memory dump at address 0x0000001A.
 - A yellow callout bubble points to the `ALIGN` directive in the assembly code, stating: "Skip one byte to address 0x40".

Data Definition Directives

The screenshot shows the uVision4 IDE with the following components:

- Registers Panel:** Shows the current state of ARM registers. R0 is 0x000000FC, R1 is 0xFFFFF04, R2 is 0x000000F0, R3 is 0xFFFFF10, R4 is 0x00000000, R5 is 0x00000000, R6 is 0x00000000, R7 is 0x00000000, R8 is 0x00000000, R9 is 0x00000000, R10 is 0x00000000, R11 is 0x00000000, R12 is 0x00000000, R13 (SP) is 0x00000000, R14 (LR) is 0x00000000, R15 (PC) is 0x00000010, CPSR is 0x000000D3, and SPSR is 0x00000000.
- Disassembly Panel:** Shows assembly instructions. The instruction at address 0x00000040 is highlighted: `ANDEQ R0, R0, R0`.
- Source File Panel:** Shows the assembly code in `ex1.asm`. The code defines variables and data:


```

12
13 two    DCW    2
14
15
16
17 four   &      4, 4
18
19         DCD    2_1010 ; Binary
20         DCD    -2_1010 ; Binary
21         DCD    8_12345670 ; Octal
22         DCD    -8_12345670 ; Octal
23
24         DCB    1      ; Any data
25 data_1  SPACE 5      ; reserves a ZEROED 5 bytes block of me
26 data_2  %          5 ; the "%" here means SPACE
27         ALIGN
28
29 data_3  SPACE 5
30         END
      
```
- Memory Panel:** Shows a memory dump starting at address 0x00000000. The dump is organized into columns for the 1st, 2nd, 3rd, and 4th bytes of each word. Annotations point to specific values:

Addresses	1st byte	2nd byte	3rd byte	4th byte
0x0000	0xE3	0xA0	0x00	0xFC
0x0004	0xE3	0xE0	0x10	0xFB
0x0008	0xE3	0xA0	0x20	0xF0
0x000C	0xE3	0xE0	0x30	0xEF
0x0010	0xEA	0xFF	0xFF	0xFE
0x0014	0x01	0x01	0x01	0x01
0x0018	0x41	0x00	0x00	0x02
0x001C	0x00	0x00	0x00	0x04
0x0020	0x00	0x00	0x00	0x04
0x0024	0x00	0x00	0x00	0x0A
0x0028	0xFF	0xFF	0xFF	0xF6
0x001C	0x00	0x29	0xCB	0xB8
0x0030	0xFF	0xD6	0x34	0x48
0x0034	0x01	0x00	0x00	0x00
0x0038	0x00	0x00	0x00	0x00
0x003C	0x00	0x00	0x00	0x00
0x0040	0x00	0x00	0x00	0x00
0x0044	0x00	0x00	0x00	0x00
0x0048				
- Command Panel:** Shows the status of the simulation: `*** Restricted Version with 32768 Byte Code Size Limit` and `*** Currently used: 72 Bytes (0%)`.

Annotations in the image include:

- loop = 0x10**: Points to the value 0x10 in the memory dump at address 0x0004.
- one = 0x14**: Points to the value 0x14 in the memory dump at address 0x0008.
- Letter = 0x18**: Points to the value 0x18 in the memory dump at address 0x000C.
- two = 0x1A**: Points to the value 0x1A in the memory dump at address 0x0018.
- four = 0x1C**: Points to the value 0x1C in the memory dump at address 0x001C.
- data_1 = 0x35**: Points to the value 0x35 in the memory dump at address 0x0024.
- data_2 = 0x3A**: Points to the value 0x3A in the memory dump at address 0x0028.
- data_3 = 0x30**: Points to the value 0x30 in the memory dump at address 0x0030.
- These are 72 bytes.**: A blue cloud-shaped annotation pointing to the memory dump area.