

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 window 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

The screenshot shows the uVision4 IDE interface. The 'Debug' menu is open, displaying options such as 'Start/Stop Debug Session' (Ctrl+F5), 'Reset CPU', 'Run' (F5), 'Stop', 'Step' (F11), 'Step Over' (F10), 'Step Out' (Ctrl+F11), 'Run to Cursor Line' (Ctrl+F10), 'Show Next Statement', 'Breakpoints...' (Ctrl+B), 'Insert/Remove Breakpoint' (F9), 'Enable/Disable Breakpoint' (Ctrl+F9), 'Disable All Breakpoints', 'Kill All Breakpoints' (Ctrl+Shift+F9), 'OS Support', 'Execution Profiling', 'Memory Map...', 'Inline Assembly...', and 'Function Editor (Open Ini File)...'.

The main editor window displays assembly code for data definitions:

```

data_definitions, CODE, READONLY
;FC; Store a Positive HEX number in r0
;FC; Store a negative HEX number in r1
;40; Store a Positive decimal number in r2
;40; Store a negative decimal number in r3

1 ; the "=" here means DCB
; 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.
; 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
; the "&" here means DCD
; DCD must start at a multiple of 4 address location
0 ; Binary positive number
0 ; Binary negative number
;45670 ; Octal positive number
;45670 ; Octal negative number

; Any data directive can be without label
; reserves a ZEROED 5 bytes block of memory
25 data_1 SPACE 5
26 data_2 % 5
27 ALIGN
28
29 data_3 SPACE 5

```

The 'Build Output' window 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).

```

The status bar at the bottom indicates 'Simulation' mode, 'L:13 C:63', and 'CAP NUM'.

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 main workspace is divided into several panes:

- Registers:** A table showing the current state of ARM registers. R0 through R15 are all at 0x00000000. CPSR is 0x000000D3 and SPSR is 0x00000000.
- Disassembly:** Shows the assembly code being executed. The current instruction is `MOV r0, # 0xFC` at address 0x00000000.
- ex1.asm:** The source assembly file. It contains several data definition directives:


```

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
      
```
- Command:** Displays the status: "Restricted Version with 32768 Byte Code Size Limit" and "Currently used: 72 Bytes (0%)".
- Call Stack + Locals:** Shows a single entry: `_asm_0x0` at location 0x00000000, type `void f()`.

The bottom status bar indicates the simulation is running, with a timer at 0.00000000 sec and L3 C:1.

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 a data definition section using `AREA`, `ENTRY`, and `DCD` directives. The `Memory Windows` pane on the left is expanded, showing a list of memory windows (Memory 1, Memory 2, Memory 3, Memory 4). The `Command` window at the bottom shows the status of the simulation, including the code size limit and the current usage.

Assembly Code:

```

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

```

Memory 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
        &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.
        ; 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
16
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

```

Command Window:

```

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

```

Call Stack + Locals:

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

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

Data Definition Directives

The screenshot shows the uVision4 IDE with the following components:

- Registers:** A list of registers (R0-R15, CPSR, SPSR) with their current values. R0 is highlighted with a value of 0x00000000.
- Disassembly:** A list of assembly instructions. The instruction at address 0x00000000 is highlighted: `MOV r0, # 0xFC; Store a Positive HEX number in r0`. The instruction at address 0x00000004 is also highlighted: `MOV r1, #-0xFC; Store a negative HEX number in r1`.
- Assembly Source:** The source code for `ex1.asm` is shown. It includes directives like `AREA More_data_definitions, CODE, READONLY`, `ENTRY`, and various `MOV`, `DCB`, `DCW`, and `DCD` instructions.
- Memory:** A memory dump showing the first 48 bytes of memory. The first four bytes are `E3 A0 00 FC`, which correspond to the first instruction in the disassembly.

A blue cloud bubble with the text "Press Step, or F11" points to the assembly code.

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; The first character in the string is 'A'
11 ; The second character is 'a' after
12 ; DCB can be used to define a 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

```

Memory1

Address	0	4	8	12	16	20	24	28	32	36	40	44	48
0x00000000	E3	A0	00	FC	E3	E0	10	FB	E3	A0	20	F0	E3
0x00000004	E0	30	EF	EA	FF	FF	FE	01	01	01	01	41	00
0x00000008	00	02	00	00	00	04	00	00	00	00	00	00	04
0x0000000C	00	00	00	0A	FF	FF	FF	F6	00	29	CB	B8	FF
0x00000010	D6	34	48	01	00	00	00	00	00	00	00	00	00
0x00000014	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000018	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0000001C	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000020	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
0x00000028	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0000002C	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000030	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000034	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000038	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0000003C	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000040	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000044	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

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 ; The first four bytes of the data in memory
10 Letter DCB  &41 ; The first byte of the data in memory after
11                ; DCB can store data at any memory location.
12                ; Must start at an even address location.
13 two    DCW  2 ; Must start at an even address location. One byte to be skipped to adjust the
14                ; IF YOU PUT ALIGN BEFORE THIS DCW, IT
15                ; TO MAKE THE ADDRESS MULTIPLE OF 4
16 four   & 4,4 ; the "&" here means DCD
17                ; DCD must start at a multiple of 4 address
18                ; Binary positive number
19 DCD 2_1010 ; Binary positive number

```

Memory 1

Address	0	4	8	12	16	20	24	28	32	36	40	44	48	
0x00000000	E3	A0	00	FC	E3	E0	10	FB	E3	A0	20	F0	E3	E0
0x00000004	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000008	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0000000C	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000014	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000018	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0000001C	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000020	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
0x00000028	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0000002C	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000034	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000038	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0000003C	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00000044	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

Command

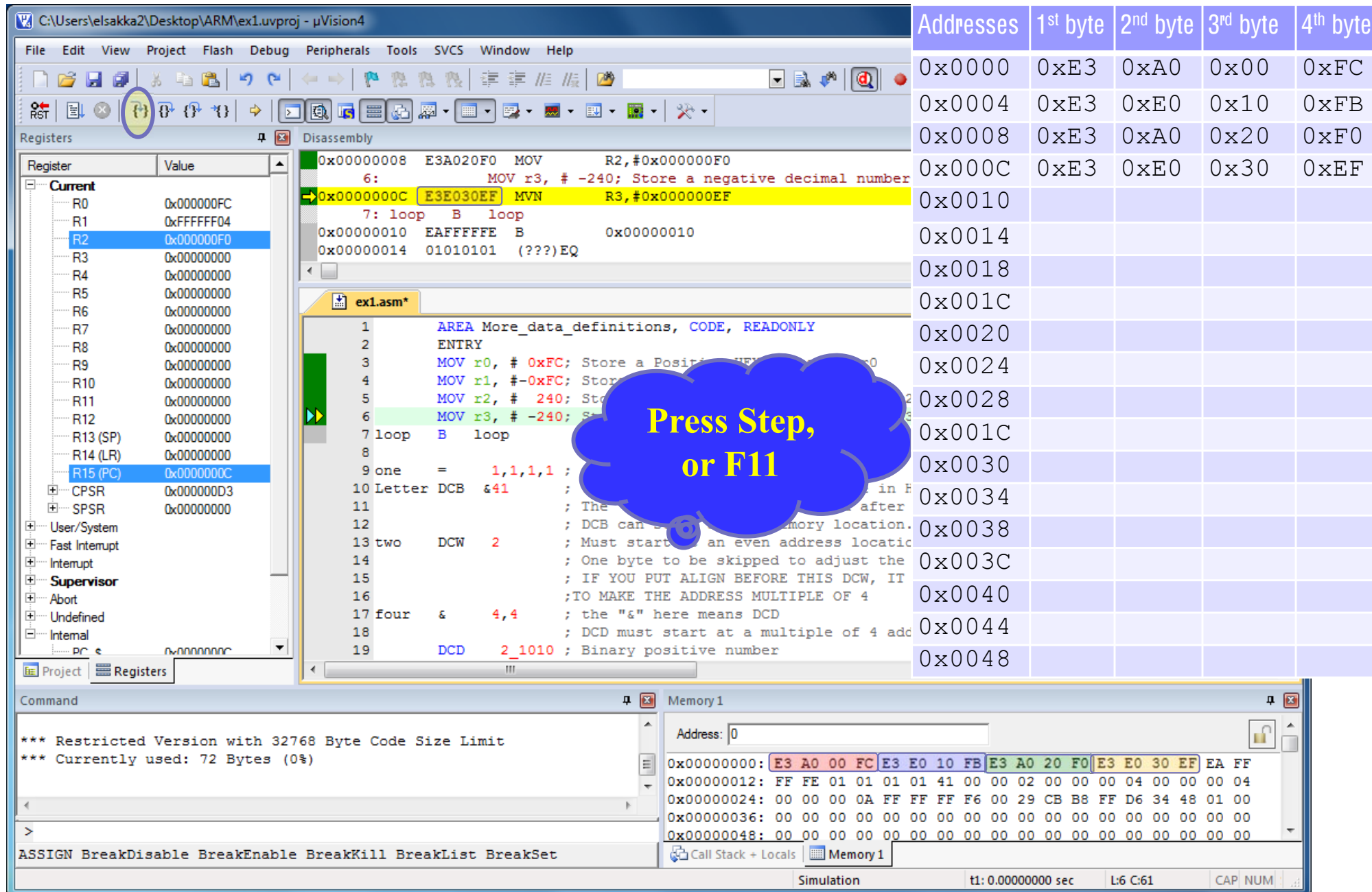
```

*** 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 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 the current register values. The main window displays the Disassembly and Source code windows. The Disassembly window shows the following instructions:

```

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

```

The Source code window shows the following assembly code:

```

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 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 add
19 DCD 2_1010; Binary positive number

```

A blue cloud bubble with the text "Press Step, or F11" points to the assembly window. The bottom pane shows the Command window with the following text:

```

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

```

The Memory window shows the following memory dump:

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				
0x00000014				
0x00000018				
0x0000001C				
0x00000020				
0x00000024				
0x00000028				
0x0000001C				
0x00000030				
0x00000034				
0x00000038				
0x0000003C				
0x00000040				
0x00000044				
0x00000048				

Data Definition Directives

loop = 0x10

Press Step, or F11

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				
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 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 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
  
```

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

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet

Simulation t1: 0.00000000 sec L7 C:16 CAP NUM

Data Definition Directives

The screenshot shows the uVision4 IDE with the following components:

- Registers:** R0 (0x000000FC), R1 (0xFFFFF04), R2 (0x000000F0), R3 (0xFFFFF10), 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 EAEFFFE B 0x00000010
0x00000014 01010101 (???)EQ

```
- Assembly Source (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 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

```
- Memory Dump:**

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				
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:**

```

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

```


Data Definition Directives

The "=" here means DCB

loop = 0x10

one = 0x14

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				
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
12
13 two DCW 2; Must start at an even address locatio
14
15
16
17 four & 4,4; the "&" here means DCD
18
19 DCD 2_1010; DCD must start at a multiple of 4 add; Binary positive number
  
```

Simulation

t1: 0.00000000 sec L:9 C:46 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.
- Disassembly:** A list of instructions with their addresses and values.
- Assembly Source (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
11 ; The "0x" prefix is NOT allowed after DCB can start at any memory location.
12 ; Must start at an even address location
13 two DCW 2 ; One byte to be skipped to adjust the
14 ; IF YOU PUT ALIGN BEFORE THIS DCW, IT
15 ; TO MAKE THE ADDRESS MULTIPLE OF 4
16 four & 4,4 ; the "&" here means DCD
17 ; DCD must start at a multiple of 4 address
18 DCD 2_1010 ; Binary positive number
19

```
- Memory Window:** A table showing memory addresses and their corresponding byte values.

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 0x0000000C.
- Letter = 0x18:** Points to the value 0x18 in the memory dump at address 0x00000010.
- The "&" means a HEX number:** A green callout pointing to the "&41" in the assembly code.

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			
0x001C				
0x0020				
0x0024				
0x0028				
0x001C				
0x0030				
0x0034				
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				

Data Definition Directives

Annotations:

- loop = 0x10** (points to address 0x00000010)
- one = 0x14** (points to address 0x00000014)
- Letter = 0x18** (points to address 0x00000018)
- two = 0x1A** (points to address 0x0000001A)

Must start at an even address location

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				
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 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: *** 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 L:9 C:46 CAP NUM

Data Definition Directives

Annotations:

- loop = 0x10**: Points to the value 0x10 in the assembly code.
- one = 0x14**: Points to the value 0x14 in the assembly code.
- Letter = 0x18**: Points to the value 0x18 in the assembly code.
- two = 0x1A**: Points to the value 0x1A in the assembly code.
- four = 0x1C**: Points to the value 0x1C in the assembly code.
- The "&" here means DCD**: Points to the ampersand in the assembly code.

Assembly Code (ex.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

```

Memory Dump (Memory1):

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				
0x0028				
0x001C				
0x0030				
0x0034				
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				

Simulation Status: Simulation, t1: 0.00000000 sec, L:9 C:46, CAP: NUM

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 code. The right pane shows the Memory window with a hex dump. Annotations highlight specific data definitions and their corresponding memory addresses.

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 0x0014.
- four = 0x1C**: Points to the value 0x1C in the memory dump at address 0x0018.

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 ALIGNMENT 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 list. The main pane shows the Disassembly view of the assembly file `ex1.asm`. The right pane shows a memory dump.

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 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:

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				
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	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 the Memory window with a hex dump. Annotations with green boxes and arrows point to specific values in the code and memory:

- 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.

The assembly code in the Disassembly window is as follows:

```

0x0000001C 00000004 ANDEQ R0,R0,R10
0x00000020 00000004 ANDEQ R0,R0,R10
0x00000024 0000000A ANDEQ R0,R0,R10
0x00000028 FFFFFFFF6 (???)
0x0000002C 0029CBB8 (???) EQ
0x00000030 FFD63448 (???)

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

The memory dump in the Memory window shows the following hex values:

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				
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 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 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 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				
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 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, and SPSR is 0x00000000.
- Disassembly Panel:** Shows the assembly code being executed. The instruction at address 0x00000030 is highlighted: `0x00000030 FFD63448 (???) EQ`.
- Source File Panel (ex1.asm):** Shows the assembly source code with annotations:
 - `two DCW 2`: Annotated with `two = 0x1A`.
 - `four & 4,4`: Annotated with `four = 0x1C`.
 - `DCD 2_1010`: Annotated with `Letter = 0x18`.
 - `DCD -2_1010`: Annotated with `one = 0x14`.
 - `DCD 8_12345670`: Annotated with `loop = 0x10`.
 - `DCD -8_12345670`: Annotated with `two = 0x1A`.
- 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. The values are:

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				
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				
- Command Panel:** Shows the status of the simulation: `*** Restricted Version with 32768 Byte Code Size Limit` and `*** Currently used: 72 Bytes (0%)`.
- Simulation Panel:** Shows the current simulation time: `tl: 0.00000000 sec` and `L9 C:46`.

Data Definition Directives

The screenshot displays the uVision4 IDE interface. The main window shows the assembly code for 'ex1.asm'. The code defines several data items using directives: 'two' (DCW, 2 words), 'four' (DCB, 4 bytes), and 'data_1', 'data_2', 'data_3' (SPACE, 5 bytes each). The code also includes an 'ALIGN' directive. The 'Registers' window on the left shows the current state of the ARM registers. The 'Memory' window at the bottom shows the memory dump starting at address 0x00000000.

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.

Memory Dump (Addresses 0x00000000 to 0x00000048):

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				
0x0038				
0x003C				
0x0040				
0x0044				
0x0048				

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 A MULTIPLE 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
  
```

Registers:

Register	Value
R0	0x000000FC
R1	0xFFFFFFFF
R2	0x000000F0
R3	0xFFFFFFFF
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

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 00 02 00	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 Status: Simulation, t1: 0.00000000 sec, L9 C:46, CAP NUM

Data Definition Directives

The screenshot displays the uVision4 IDE interface. The main window shows the disassembly of an ARM program. The registers window on the left shows the current state of the registers. The command window at the bottom shows the execution status. The memory window at the bottom right shows the memory 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 0x00000010.
- four = 0x1C**: Points to the value 0x1C in the memory dump at address 0x00000014.

Assembly Code:

```

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 A MULTIPLE OF 4
17 four   &      4,4    ; the "&" here means PC
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:

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			
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 00 02 00	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

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 0x0014.
- four = 0x1C**: Points to the value 0x1C in the memory dump at address 0x0018.
- 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 A 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
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: C46, CAP NUM

Data Definition Directives

Registers

Register	Value
R0	0x000000FC
R1	0xFFFFFFFF
R2	0x000000F0
R3	0xFFFFFFFF
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

```

0x0000002C 0029CBB8 (???) EQ
0x00000030 FFD63448 (???)
0x00000034 01000000 (???) EQ
0x00000038 00000000 ANDEQ R0,R0,R0
0x0000003C 00000000 ANDEQ R0,R0,R0
0x00000040 00000000 ANDEQ R0,R0,R0
  
```

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 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

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	0x00	0x00	0x00
0x0038	0x00	0x00	0x00	0x00
0x003C	0x00	0x00	0x00	0x00
0x0040				
0x0044				
0x0048				

Call Stack + Locals

Simulation t1: 0.00000000 sec L9 C:46 CAP NUM

The "%" here means SPACE

Data Definition Directives

The screenshot displays the uVision4 IDE with the following components:

- Registers Panel:** Shows the current state of ARM registers (R0-R15, CPSR, SPSR).
- Disassembly Panel:** Shows the assembly code being executed, including instructions like `ANDEQ`.
- Source File Panel:** Shows the assembly source file `ex1.asm` with various data definition directives.
 - `two DCW 2`: Defines a word (2 bytes) at an even address.
 - `four & 4,4`: Defines a word (2 bytes) at an address aligned to 4 bytes.
 - `DCD 2_1010`, `DCD -2_1010`, `DCD 8_12345670`, `DCD -8_12345670`: Define double words (4 bytes) in binary, binary negative, octal positive, and octal negative formats.
 - `DCB 1`: Define a byte (1 byte).
 - `data_1 SPACE 5`, `data_2 % 5`: Reserve space for data.
 - `ALIGN`: Advance the location counter to the next multiple of 4.
 - `data_3 SPACE 5`: Reserve space for data.
 - `END`: End of the assembly file.
- Memory Panel:** Shows the memory dump starting at address 0x00000000. The dump is color-coded to show the layout of the data defined in the source file.

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	0x00	0x00	0x00
0x0038	0x00	0x00	0x00	0x00
0x003C	0x00	0x00	0x00	0x00
0x0040				
0x0044				
0x0048				

Annotations:

- loop = 0x10**: Points to the 3rd byte of the first word (0x0010).
- one = 0x14**: Points to the 4th byte of the first word (0x0014).
- Letter = 0x18**: Points to the 1st byte of the second word (0x0018).
- two = 0x1A**: Points to the 2nd byte of the second word (0x001A).
- four = 0x1C**: Points to the 4th byte of the second word (0x001C).
- data_1 = 0x35**: Points to the 1st byte of the first double word (0x0030).
- data_2 = 0x3A**: Points to the 2nd byte of the first double word (0x003A).
- Skip one byte to address 0x40**: Points to the start of the next word (0x0040).

Data Definition Directives

The screenshot shows the uVision4 IDE with the following components:

- Registers Panel:** Shows the current state of ARM registers (R0-R15, CPSR, SPSR).
- Disassembly Panel:** Shows the assembly code being executed, including the instruction `ANDEQ R0, R0, R0` at address 0x00000040.
- Source File Panel (ex.asm):** Contains the assembly code with data definition directives:


```

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 "%"
27         ALIGN  ; ADVANCE
28
29 data_3  SPACE 5 ; OF 4 ADDRESS LOCATION
30         END
      
```
- Memory Panel:** Shows the memory layout starting at address 0x00000000. The data is organized into 4-byte words, with some bytes highlighted in red to indicate 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, including the message "Currently used: 72 Bytes (0%)".

Callouts and Annotations:

- loop = 0x10:** Points to the value 0x10 in the memory layout at address 0x0004.
- one = 0x14:** Points to the value 0x14 in the memory layout at address 0x0008.
- Letter = 0x18:** Points to the value 0x18 in the memory layout at address 0x000C.
- two = 0x1A:** Points to the value 0x1A in the memory layout at address 0x0018.
- four = 0x1C:** Points to the value 0x1C in the memory layout at address 0x001C.
- data_1 = 0x35:** Points to the value 0x35 in the memory layout at address 0x0030.
- data_2 = 0x3A:** Points to the value 0x3A in the memory layout at address 0x0034.
- data_3 = 0x30:** Points to the value 0x30 in the memory layout at address 0x0038.
- These are 72 bytes:** A blue cloud-shaped callout pointing to the memory layout from address 0x0018 to 0x0048.