

These slides are being provided with permission from the copyright for CS2208 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 09:

# ARM Pseudo Instructions

*Computer Science Department*

*CS2208: Introduction to Computer Organization and Architecture*

*Winter 2021-2022*

*Instructor: Mahmoud R. El-Sakka*

*Office: MC-419*

*Email: [elsakka@csd.uwo.ca](mailto:elsakka@csd.uwo.ca)*

*Phone: 519-661-2111 x86996*

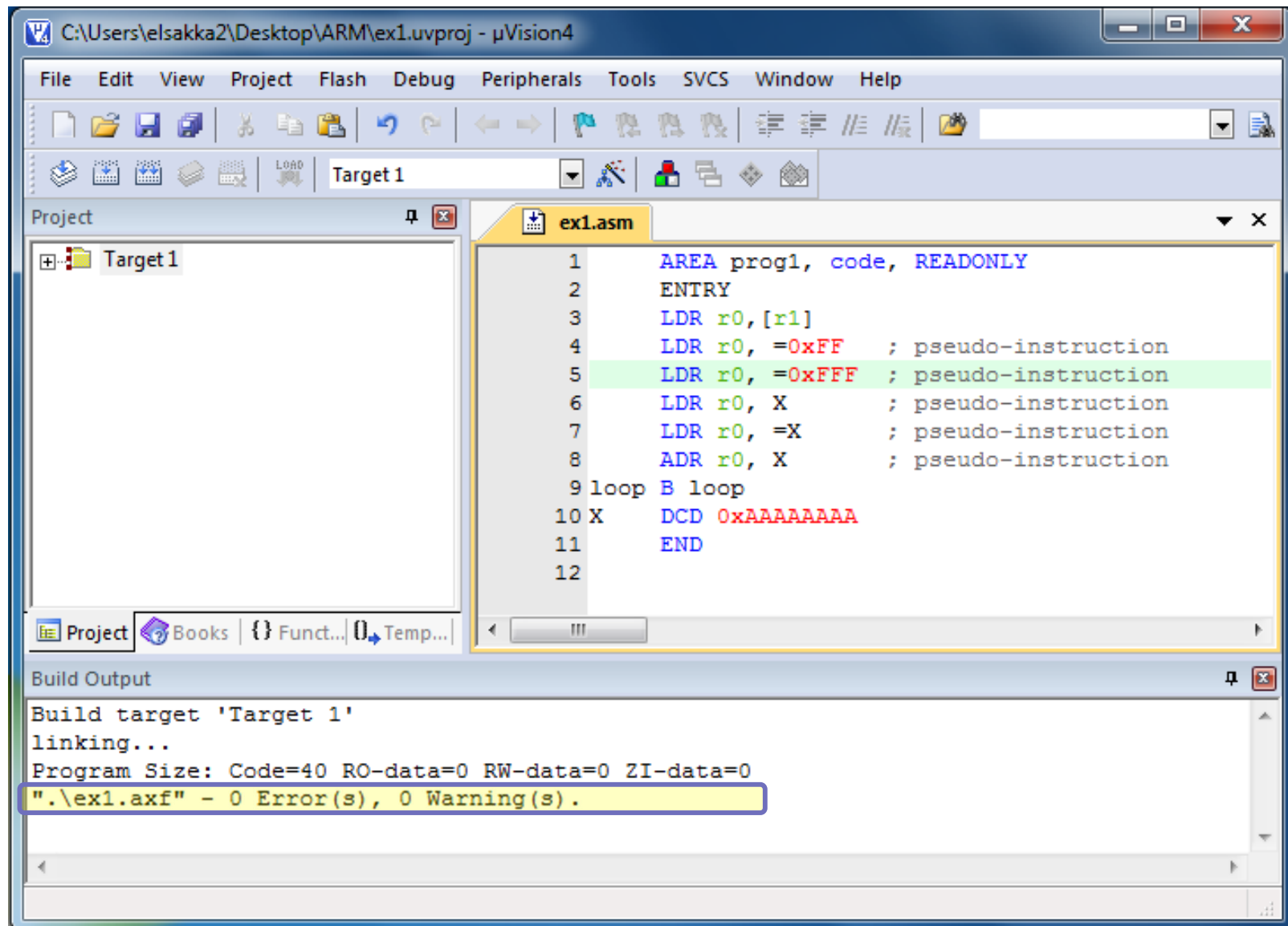
# ARM pseudo-instructions

- The ARM assembler supports several *pseudo instructions* that are translated into the appropriate combination of ARM words at assembly time.
- Consider the following assembly program:

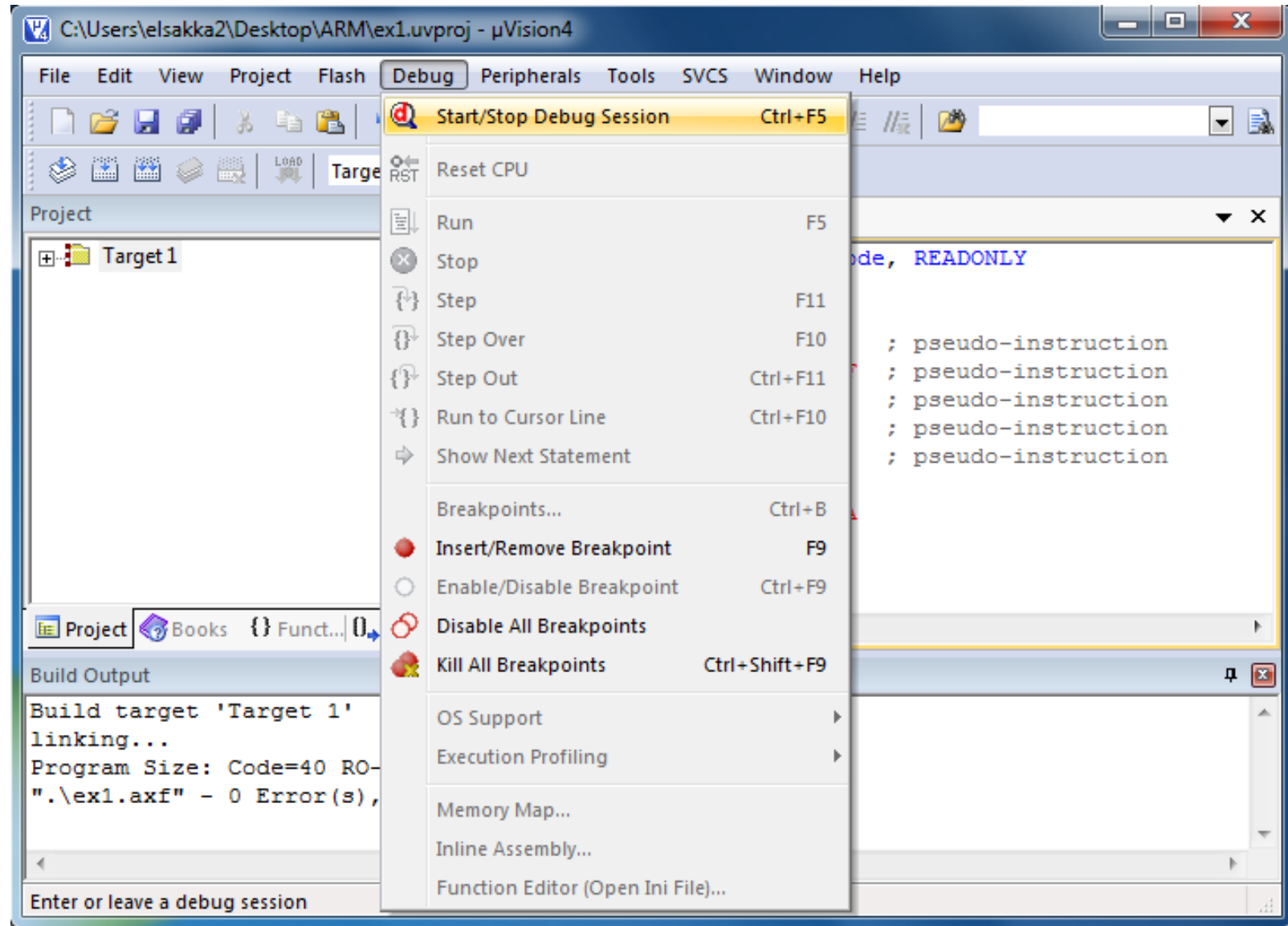
```
AREA prog1, code, READONLY
ENTRY
LDR r0, [r1]
LDR r0, =0xFF      ; pseudo-instruction
LDR r0, =0xFFF     ; pseudo-instruction
LDR r0, X          ; pseudo-instruction
LDR r0, =X         ; pseudo-instruction
ADR r0, X          ; pseudo-instruction

loop B loop
X    DCD 0xAAAAAAAA
END
```

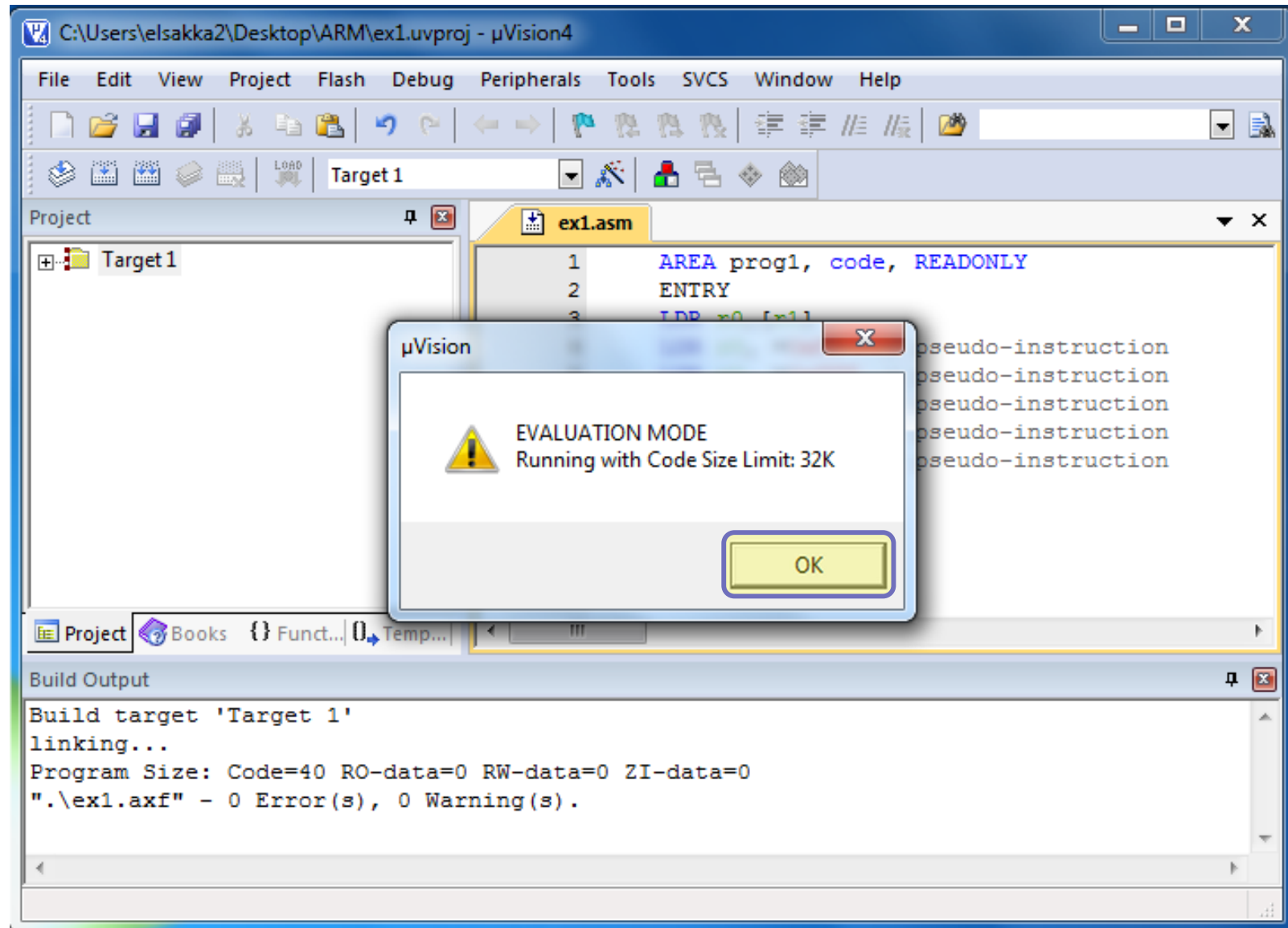
# ARM pseudo-instructions



# ARM pseudo-instructions



# ARM pseudo-instructions



# ARM pseudo-instructions

The screenshot displays the µVision4 IDE interface. The **Disassembly** window shows the following instructions:

Address	Hex	Mnemonic	Comment
3:	E5910000	LDR	R0, [R1]
4:	E3A000FF	MOV	R0, #0x000000FF ; pseudo-instruction
5:	E59F0010	LDR	R0, [PC, #0x0010] ; pseudo-instruction
6:	E59F0008	LDR	R0, [PC, #0x0008] ; pseudo-instruction
7:	E59F000C	LDR	R0, [PC, #0x000C] ; pseudo-instruction
8:	E28F0000	ADD	R0, PC, #0x00000000 ; pseudo-instruction
9:		loop B loop	
10:	EAF00018	B	0x00000018
11:	AAAAAAAC	BGE	0xFEAAAAAC
12:	000000FF	???	EQ
13:	0000001C	ANDEQ	R0, R0, R12, LSL R0
14:	00000000	ANDEQ	R0, R0, R0

The **Registers** window shows the current state of registers R0 through R15, CPSR, and SPSR. The **Source** window shows the assembly code for `ex1.asm`:

```

1  AREA prog1, code, READONLY
2  ENTRY
3  LDR r0, [r1]
4  LDR r0, =0xFF ; pseudo-instruction
5  LDR r0, =0xFFFF ; pseudo-instruction
6  LDR r0, X ; pseudo-instruction
7  LDR r0, =X ; pseudo-instruction
8  ADR r0, X ; pseudo-instruction
9  loop B loop
10 X DCD 0xAAAAAA
11 END
12

```

Press Step,  
or F11

# ARM pseudo-instructions

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:**

Register	Value
<b>R0</b>	<b>0xE5910000</b>
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
<b>R15 (PC)</b>	<b>0x00000004</b>
CPSR	0x000000D3
SPSR	0x00000000
- Disassembly Window:**

```

3:      LDR r0,[r1]
0x00000000 E5910000 LDR      R0,[R1]
4:      LDR r0,=0xFF ; pseudo-instruction
0x00000004 E3A000FF MOV      R0,#0x000000FF
5:      LDR r0,=0xFFF ; pseudo-instruction
0x00000008 E59F0010 LDR      R0,[PC,#0x0010]
6:      LDR r0,X ; pseudo-instruction
0x0000000C E59F0008 LDR      R0,[PC,#0x0008]
7:      LDR r0,=X ; pseudo-instruction
0x00000010 E59F000C LDR      R0,[PC,#0x000C]
8:      ADR r0,X ; pseudo-instruction
0x00000014 E28F0000 ADD      R0,PC,#0x00000000
9:      loop B loop
0x00000018 EAffffff B        0x00000018
0x0000001C AAAAAAAA BGE     0xFEAAAAACC
0x00000020 0000FFFF ???EQ
0x00000024 0000001C ANDEQ   R0,R0,R12,LSL R0
0x00000028 00000000 ANDEQ   R0,R0,R0

```
- Source Window (ex1.asm):**

```

1      AREA prog1, code, READONLY
2      ENTRY
3      LDR r0,[r1]
4      LDR r0,=0xFF ; pseudo-instruction
5      LDR r0,=0xFFF ; pseudo-instruction
6      LDR r0,X ; pseudo-instruction
7      LDR r0,=X ; pseudo-instruction
8      ADR r0,X ; pseudo-instruction
9      loop B loop
10     X      DCD 0xAAAAAAAA
11
12

```

Press Step,  
or F11



# ARM pseudo-instructions

When executing the instruction at location 0x00000008, the PC value will be 0x00000010

How is this offset calculated?

Press Step, or F11

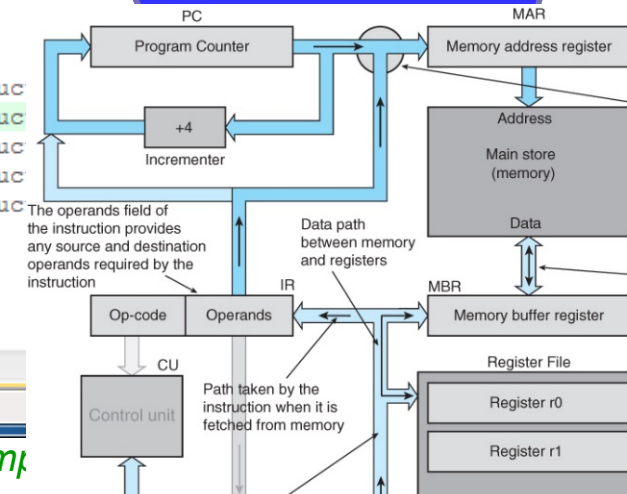
The screenshot shows the uVision4 IDE with the Disassembly window and the source file 'ex1.asm'. The Disassembly window displays the following instructions:

Address	Disassembly	Comment
0x00000000	E5910000 LDR R0, [R1]	
0x00000004	E3A000FF MOV R0, #0x000000FF	; pseudo-instruction
0x00000008	E59F0010 LDR R0, [PC, #0x0010]	; pseudo-instruction
0x0000000C	E59F0008 LDR R0, [PC, #0x0008]	; pseudo-instruction
0x00000010	E59F000C LDR R0, [PC, #0x000C]	; pseudo-instruction
0x00000014	E28F0000 ADD R0, PC, #0x00000000	
0x00000018	EAFFFFFE B 0x00000018	
0x0000001C	AAAAAAABGE 0xFEAAACC	
0x00000020	00000FFF ???EQ	
0x00000024	0000001C ANDEQ R0, R0, R12, LSL R0	
0x00000028	00000000 ANDEQ R0, R0, R0	

The source file 'ex1.asm' shows the following code:

```

1 AREA prog1, code, READONLY
2 ENTRY
3 LDR r0, [r1]
4 LDR r0, =0xFF ; pseudo-instruction
5 LDR r0, =0xFFFF ; pseudo-instruction
6 LDR r0, X ; pseudo-instruction
7 LDR r0, =X ; pseudo-instruction
8 ADR r0, X ; pseudo-instruction
9 loop B loop
10 X DCD 0xAAAAAAAA
11 END
  
```





# ARM pseudo-instructions

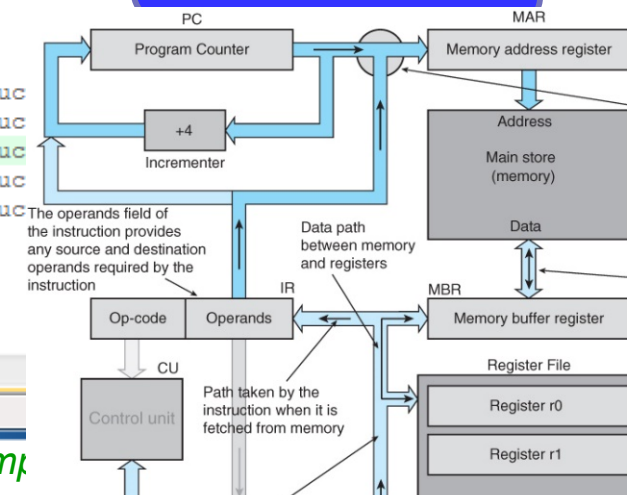
When executing the instruction at location 0x0000000C, the PC value will be 0x00000014

How is this offset calculated?

Press Step, or F11

The screenshot shows the uVision4 IDE with the following components:

- Registers Window:** Shows the PC register (R15) at address 0x0000000C.
- Disassembly Window:** Shows the instruction at address 0x0000000C: `LDR r0, X ; pseudo-instruction`. The next instruction at 0x00000014 is `ADD R0, PC, #0x00000000`.
- Source Code Window (ex1.asm):** Shows the assembly code with comments: `LDR r0, X ; pseudo-instruction` at line 6 and `DCD 0xAAAAAAAA` at line 10.



# ARM pseudo-instructions

When executing the instruction at location 0x00000010, the PC value will be 0x00000018

How is this offset calculated?

Press Step, or F11

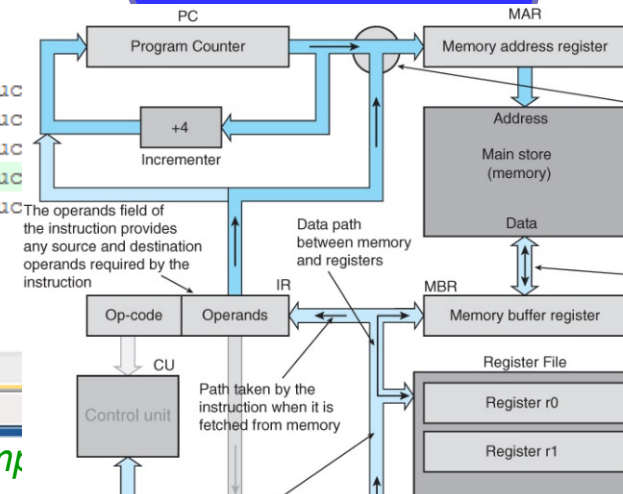
The screenshot shows the uVision4 IDE with the following components:

- Registers Window:** Shows R15 (PC) at 0x00000010.
- Disassembly Window:**
  - Address 0x00000000: LDR r0, [r1] (E5910000)
  - Address 0x00000004: LDR r0, =0xFF ; pseudo-instruction (E3A000FF)
  - Address 0x00000008: LDR r0, =0xFFFF ; pseudo-instruction (E59F0010)
  - Address 0x0000000C: LDR r0, X ; pseudo-instruction (E59F0008)
  - Address 0x00000010: LDR r0, =X ; pseudo-instruction (E59F000C) - **Current instruction**
  - Address 0x00000014: ADR r0, X ; pseudo-instruction (E28F0000)
  - Address 0x00000018: loop B loop (EAF FFFE)
  - Address 0x0000001C: BGE (AAAAAAA)
  - Address 0x00000020: ANDEQ (0000FFF)
  - Address 0x00000024: ANDEQ (000001C)
  - Address 0x00000028: ANDEQ (0000000)
- Source Window (ex1.asm):**

```

1 AREA prog1, code, READONLY
2 ENTRY
3 LDR r0, [r1]
4 LDR r0, =0xFF ; pseudo-instruc
5 LDR r0, =0xFFFF ; pseudo-instruc
6 LDR r0, X ; pseudo-instruc
7 LDR r0, =X ; pseudo-instruc
8 ADR r0, X ; pseudo-instruc
9 loop B loop
10 X DCD 0xAAAAAAAA
11 END

```



# ARM pseudo-instructions

Compare the translations of  
**LDR r0, =X**  
 and  
**ADR r0, X**

When  
 executing the  
 instruction  
 at location  
 0x00000014,  
 the PC value  
 will be  
 0x0000001C

How is this  
 offset  
 calculated?

Press Step,  
 or F11

Registers window:

Register	Value
R0	0x0000001C
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)	0x00000014
CPSR	0x000000D3
SPSR	0x00000000

Disassembly window:

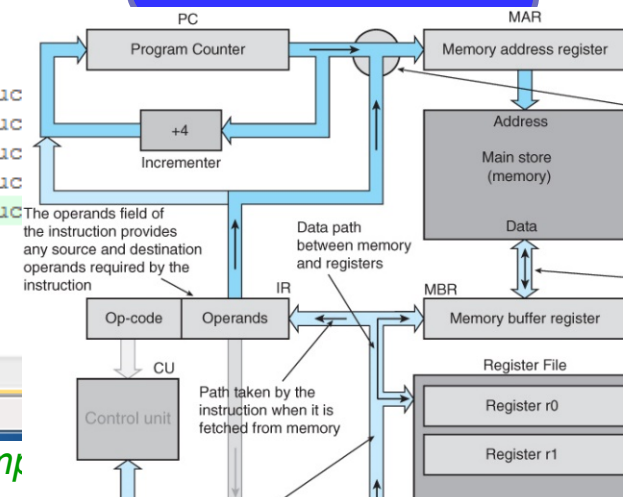
```

3:      LDR r0, [r1]
0x00000000 E5910000 LDR      R0, [R1]
4:      LDR r0, =0xFF ; pseudo-instruction
0x00000004 E3A000FF MOV      R0, #0x000000FF
5:      LDR r0, =0xFFF ; pseudo-instruction
0x00000008 E59F0010 LDR      R0, [PC, #0x0010]
6:      LDR r0, X ; pseudo-instruction
0x0000000C E59F0008 LDR      R0, [PC, #0x0008]
7:      LDR r0, =X ; pseudo-instruction
0x00000010 E59F000C LDR      R0, [PC, #0x000C]
8:      ADR r0, X ; pseudo-instruction
0x00000014 E28F0000 ADD      R0, PC, #0x00000000
9: loop B loop
0x00000018 EAFFFFFE B        0x00000018
0x0000001C AAAAAAAA BGE     0xFEAAAAAC
0x00000020 0000FFFF ???EQ
0x00000024 0000001C ANDEQ   R0, R0, R12, LSL R0
0x00000028 00000000 ANDEQ   R0, R0, R12
  
```

Source code window (ex1.asm):

```

1 AREA prog1, code, READONLY
2 ENTRY
3 LDR r0, [r1]
4 LDR r0, =0xFF ; pseudo-instruction
5 LDR r0, =0xFFF ; pseudo-instruction
6 LDR r0, X ; pseudo-instruction
7 LDR r0, =X ; pseudo-instruction
8 ADR r0, X ; pseudo-instruction
9 loop B loop
10 X DCD 0xAAAAAAAA
11 END
  
```



# ARM pseudo-instructions

Same  
address  
(no change)

The screenshot displays the uVision4 IDE interface. The **Registers** window on the left shows the current state of ARM registers, with R15 (PC) highlighted at address 0x00000018. The **Disassembly** window shows the instruction stream, with instruction 9 (a loop label) highlighted. The **ex1.asm** source window at the bottom shows the assembly code, with the loop section highlighted.

Address	Hex	Mnemonic	Comment
0x00000000	E5910000	LDR	R0, [R1]
0x00000004	E3A000FF	MOV	R0, #0x000000FF ; pseudo-instruction
0x00000008	E59F0010	LDR	R0, [PC, #0x0010] ; pseudo-instruction
0x0000000C	E59F0008	LDR	R0, [PC, #0x0008] ; pseudo-instruction
0x00000010	E59F000C	LDR	R0, [PC, #0x000C] ; pseudo-instruction
0x00000014	E28F0000	ADD	R0, PC, #0x00000000 ; pseudo-instruction
0x00000018	EAF FFFE	B	loop B loop
0x0000001C	AAAAAAA	BGE	0xFEAAAACC
0x00000020	0000FFF	???	EQ
0x00000024	000001C	ANDEQ	R0, R0, R12, LSL R0
0x00000028	0000000	ANDEQ	R0, R0, R0

```

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



# ARM pseudo-instructions

- Consider we changed the previous program as follow:

```
AREA prog1, code, READONLY
ENTRY
LDR r0, [r1]
LDR r0, =0xFF      ; pseudo-instruction
LDR r0, =0xFFF     ; pseudo-instruction
LDR r0, X          ; pseudo-instruction
LDR r0, =X         ; pseudo-instruction
ADR r0, X          ; pseudo-instruction

loop B loop
X DCD 0xAAAAAAAA
END
```

```
AREA prog1, code, READONLY
ENTRY
LDR r0, [r1]
LDR r0, =0xFF      ; pseudo-instruction
LDR r0, =0xFFF     ; pseudo-instruction
LDR r0, X          ; pseudo-instruction
LDR r0, =X         ; pseudo-instruction
ADR r0, X          ; pseudo-instruction

loop B loop

AREA prog1, data, READONLY
X DCD 0xAAAAAAAA
END
```

- What is the effect of this change on the generated code?

# ARM pseudo-instructions

Registers:

Register	Value
R0	0x0000001C
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)	0x00000018
CPSR	0x000000D3
SPSR	0x00000000

Disassembly:

```

3: LDR r0,[r1]
4: LDR r0,=0xFF ; pseudo-instruction
5: LDR r0,=0xFFFF ; pseudo-instruction
6: LDR r0,X ; pseudo-instruction
7: LDR r0,=X ; pseudo-instruction
8: ADR r0,X ; pseudo-instruction
9: loop B loop
10: DCD 0xAAAAAAAA
11: END

```

Assembly Code (ex1.asm):

```

1 AREA prog1, code, READONLY
2 ENTRY
3 LDR r0,[r1]
4 LDR r0,=0xFF ; pseudo-instruction
5 LDR r0,=0xFFFF ; pseudo-instruction
6 LDR r0,X ; pseudo-instruction
7 LDR r0,=X ; pseudo-instruction
8 ADR r0,X ; pseudo-instruction
9 loop B loop
10 DCD 0xAAAAAAAA
11 END

```

Registers:

Register	Value
R0	0x00000024
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)	0x00000018
CPSR	0x000000D3
SPSR	0x00000000

Disassembly:

```

3: LDR r0,[r1]
4: LDR r0,=0xFF ; pseudo-instruction
5: LDR r0,=0xFFFF ; pseudo-instruction
6: LDR r0,X ; pseudo-instruction
7: LDR r0,=X ; pseudo-instruction
8: ADR r0,X ; pseudo-instruction
9: loop B loop
10: DCD 0xAAAAAAAA
11: END

```

Assembly Code (ex1.asm):

```

3 LDR r0,[r1]
4 LDR r0,=0xFF ; pseudo-instruction
5 LDR r0,=0xFFFF ; pseudo-instruction
6 LDR r0,X ; pseudo-instruction
7 LDR r0,=X ; pseudo-instruction
8 ADR r0,X ; pseudo-instruction
9 loop B loop
11 AREA prog1, data, READONLY
12 DCD 0xAAAAAAAA
13 END

```

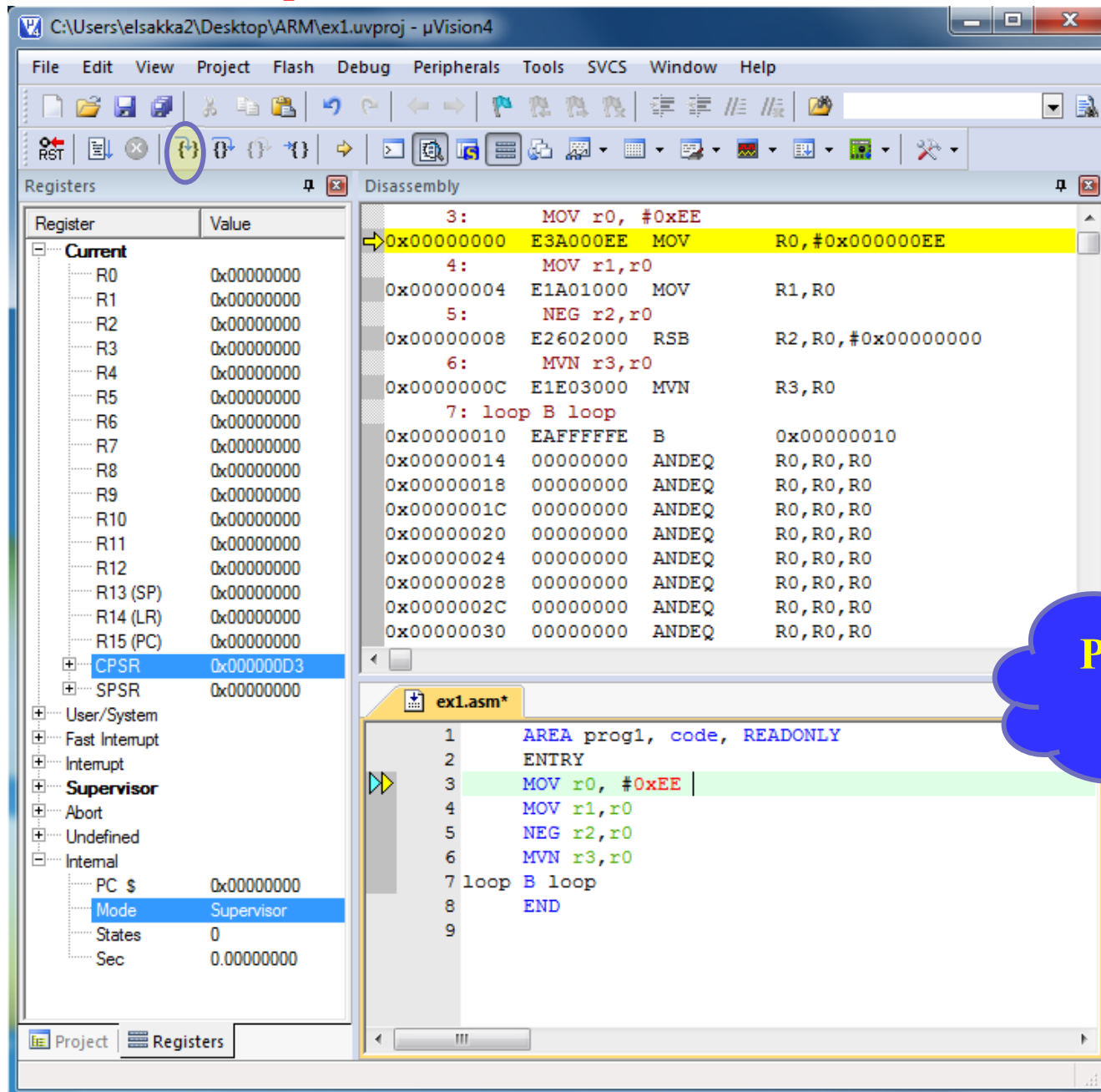


# ARM pseudo-instructions

- Consider the following assembly program:

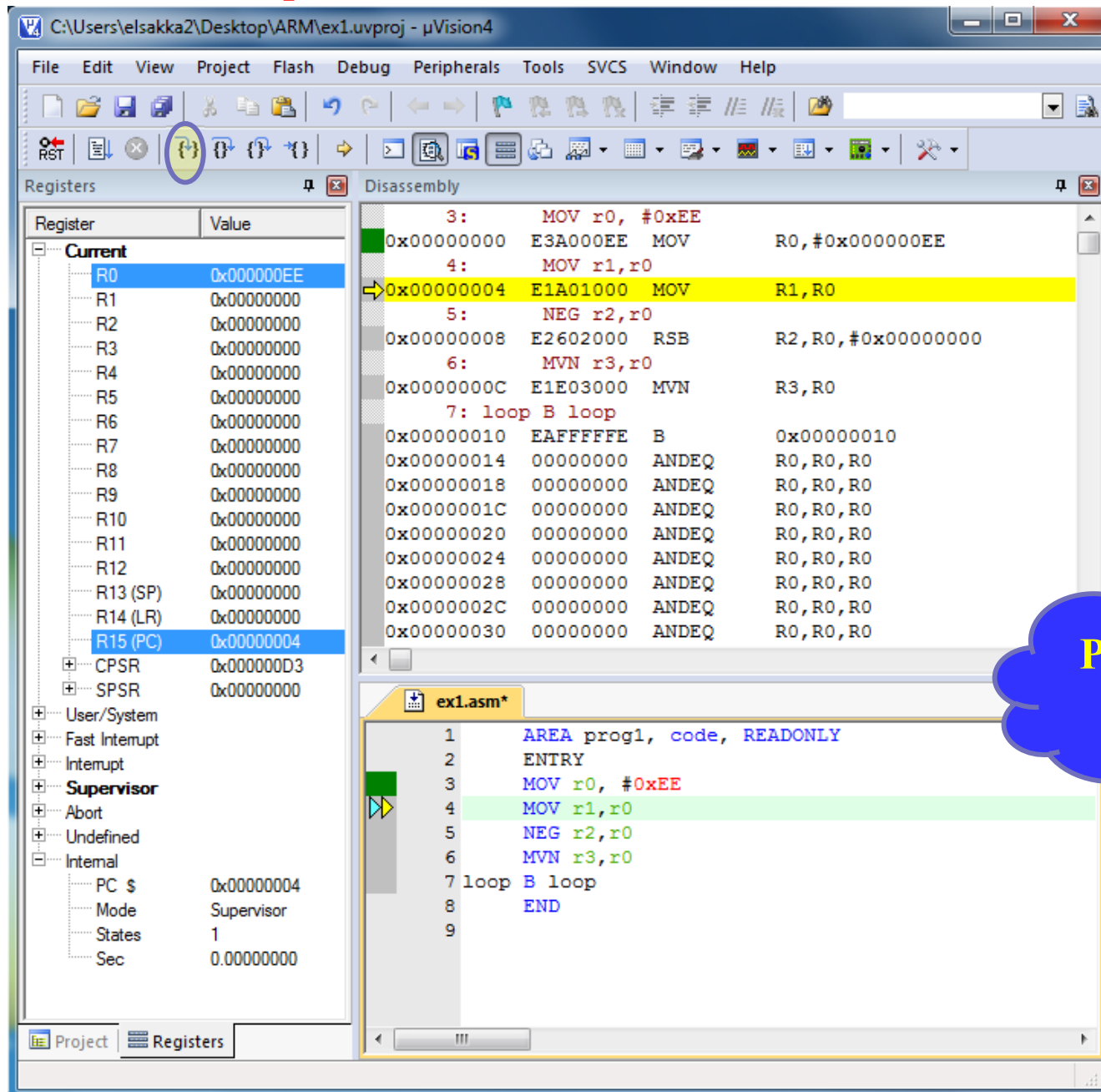
```
AREA prog1, code, READONLY
ENTRY
MOV r0, #0xEE
MOV r1, r0
NEG r2, r0
MVN r3, r0
loop B loop
END
```

# ARM pseudo-instructions



Press Step,  
or F11

# ARM pseudo-instructions



Press Step,  
or F11

# ARM pseudo-instructions

The screenshot displays the µVision4 IDE interface. The 'Registers' window on the left shows the current state of ARM registers, with R15 (PC) highlighted at address 0x00000008. The 'Disassembly' window shows the instruction stream, with the instruction 'RSB R2, R0, #0x00000000' highlighted in yellow. The 'Source' window at the bottom shows the assembly code for 'ex1.asm', with the instruction 'NEG r2, r0' highlighted in green. A blue cloud bubble on the right contains the text 'Press Step, or F11'.

Register	Value
R0	0x000000EE
R1	0x000000EE
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

```

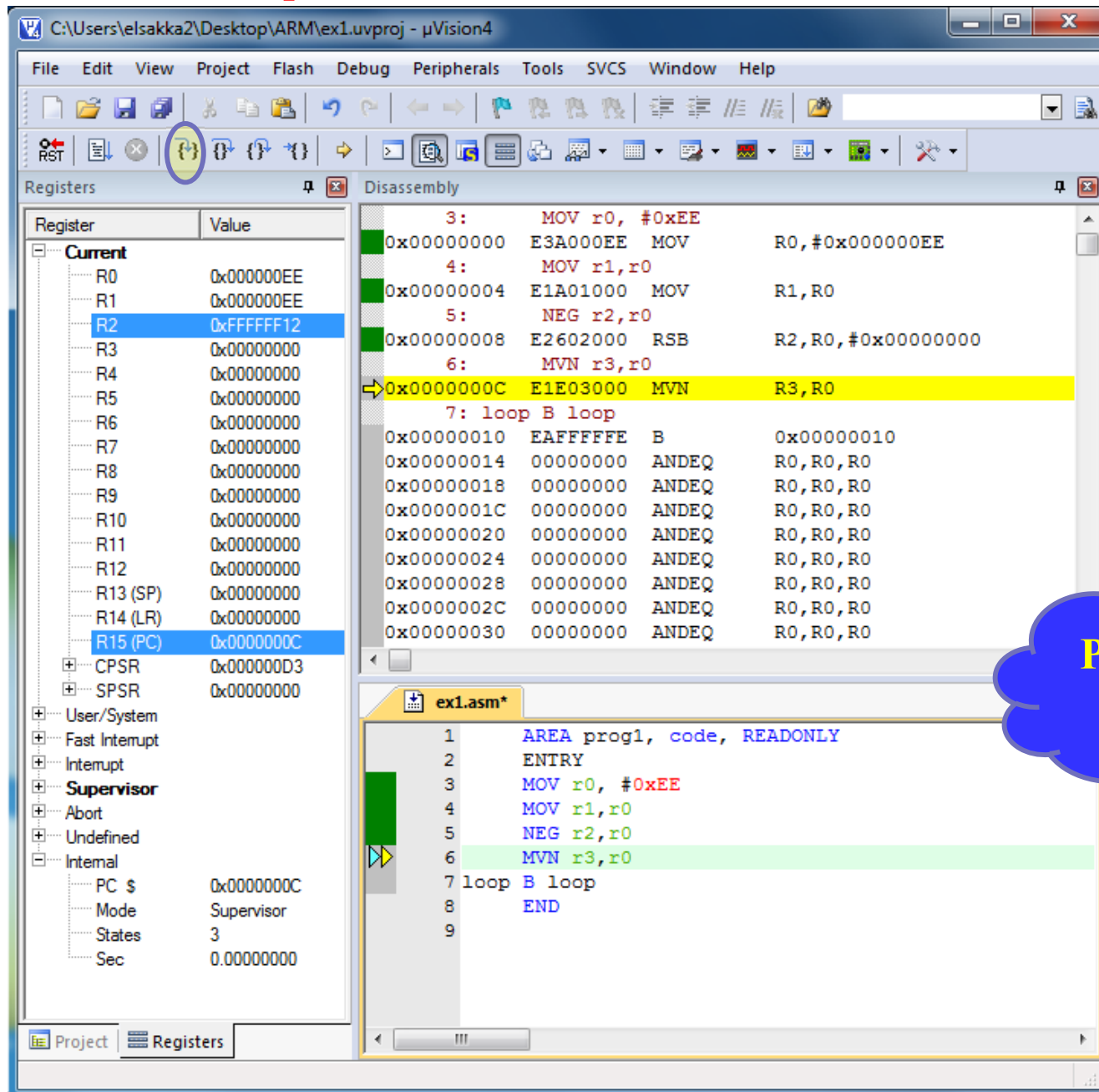
3:      MOV r0, #0xEE
0x00000000 E3A000EE MOV      R0, #0x000000EE
4:      MOV r1, r0
0x00000004 E1A01000 MOV      R1, R0
5:      NEG r2, r0
0x00000008 E2602000 RSB      R2, R0, #0x00000000
6:      MVN r3, r0
0x0000000C E1E03000 MVN      R3, R0
7: loop B loop
0x00000010 EAF000FE B        0x00000010
0x00000014 00000000 ANDEQ    R0, R0, R0
0x00000018 00000000 ANDEQ    R0, R0, R0
0x0000001C 00000000 ANDEQ    R0, R0, R0
0x00000020 00000000 ANDEQ    R0, R0, R0
0x00000024 00000000 ANDEQ    R0, R0, R0
0x00000028 00000000 ANDEQ    R0, R0, R0
0x0000002C 00000000 ANDEQ    R0, R0, R0
0x00000030 00000000 ANDEQ    R0, R0, R0
  
```

```

1  AREA prog1, code, READONLY
2  ENTRY
3  MOV r0, #0xEE
4  MOV r1, r0
5  NEG r2, r0
6  MVN r3, r0
7 loop B loop
8  END
  
```

Press Step,  
or F11

# ARM pseudo-instructions



Press Step,  
or F11

# ARM pseudo-instructions

The screenshot displays the uVision4 IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. Below the menu is a toolbar with various icons for file operations, editing, and debugging.

The **Registers** window on the left shows the current state of the ARM registers. The **Current** register set includes R0 through R15, CPSR, and SPSR. R3 is highlighted with a value of 0xFFFFFFFF. R15 (PC) is highlighted with a value of 0x00000010. The **Internal** register set shows PC \$, Mode (Supervisor), States (4), and Sec (0.00000000).

The **Disassembly** window on the right shows the assembly code being executed. The code is as follows:

```

3:      MOV r0, #0xEE
0x00000000 E3A000EE MOV      R0,#0x000000EE
4:      MOV r1,r0
0x00000004 E1A01000 MOV      R1,R0
5:      NEG r2,r0
0x00000008 E2602000 RSB      R2,R0,#0x00000000
6:      MVN r3,r0
0x0000000C E1E03000 MVN      R3,R0
7: loop B loop
0x00000010 EAF000FE B        0x00000010
0x00000014 00000000 ANDEQ    R0,R0,R0
0x00000018 00000000 ANDEQ    R0,R0,R0
0x0000001C 00000000 ANDEQ    R0,R0,R0
0x00000020 00000000 ANDEQ    R0,R0,R0
0x00000024 00000000 ANDEQ    R0,R0,R0
0x00000028 00000000 ANDEQ    R0,R0,R0
0x0000002C 00000000 ANDEQ    R0,R0,R0
0x00000030 00000000 ANDEQ    R0,R0,R0

```

The **ex1.asm\*** window at the bottom shows the source code for the assembly file:

```

1      AREA prog1, code, READONLY
2      ENTRY
3      MOV r0, #0xEE
4      MOV r1,r0
5      NEG r2,r0
6      MVN r3,r0
7 loop B loop
8      END
9

```



# ARM pseudo-instructions

- Consider we changed the previous program as follow:

```
AREA prog1, code, READONLY
ENTRY
MOV r0, #0xEE
MOV r1, r0
NEG r2, r0
MVN r3, r0
loop B loop
END
```

```
AREA prog1, code, READONLY
ENTRY
MOV r0, #-0xEE
MOV r1, r0
NEG r2, r0
MVN r3, r0
loop B loop
END      END
```

- What is the effect of this change on the generated code?

# ARM pseudo-instructions

The screenshot shows the ARM Disassembler interface. The 'Registers' pane on the left lists registers R0 through R15 (PC) and CPSR/SPSR. The 'Disassembly' pane shows the following assembly code:

```

3: MOV r0, #0xFF
4: MOV r1, r0
5: NEG r2, r0
6: MVN r3, r0
7: loop B loop
0x00000010: EAF FFFF E B 0x00000010
0x00000014: 00000000 ANDEQ R0, R0, R0
0x00000018: 00000000 ANDEQ R0, R0, R0
0x0000001C: 00000000 ANDEQ R0, R0, R0
0x00000020: 00000000 ANDEQ R0, R0, R0
0x00000024: 00000000 ANDEQ R0, R0, R0
0x00000028: 00000000 ANDEQ R0, R0, R0
0x0000002C: 00000000 ANDEQ R0, R0, R0
0x00000030: 00000000 ANDEQ R0, R0, R0

```

The 'ex1.asm' source file is shown at the bottom, containing the following code:

```

1 AREA prog1, code, READONLY
2 ENTRY
3 MOV r0, #0xFF
4 MOV r1, r0
5 NEG r2, r0
6 MVN r3, r0
7 loop B loop
8 END
9

```

The screenshot shows the ARM Disassembler interface after the execution of the instruction `MOV r0, #-0xEE`. The 'Registers' pane shows that R0 now contains the value `0xFFFFF12`. The 'Disassembly' pane shows the following assembly code:

```

3: MOV r0, #-0xEE
4: MOV r1, r0
5: NEG r2, r0
6: MVN r3, r0
7: loop B loop
0x00000010: EAF FFFF E B 0x00000010
0x00000014: 00000000 ANDEQ R0, R0, R0
0x00000018: 00000000 ANDEQ R0, R0, R0
0x0000001C: 00000000 ANDEQ R0, R0, R0
0x00000020: 00000000 ANDEQ R0, R0, R0
0x00000024: 00000000 ANDEQ R0, R0, R0
0x00000028: 00000000 ANDEQ R0, R0, R0
0x0000002C: 00000000 ANDEQ R0, R0, R0
0x00000030: 00000000 ANDEQ R0, R0, R0

```

The 'ex1.asm' source file is shown at the bottom, containing the following code:

```

1 AREA prog1, code, READONLY
2 ENTRY
3 MOV r0, #-0xEE
4 MOV r1, r0
5 NEG r2, r0
6 MVN r3, r0
7 loop B loop
8 END
9

```