

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 09:

ARM Pseudo Instructions

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 pseudo-instructions

- The ARM assembler supports a number of 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

The screenshot displays the uVision4 IDE interface. The main window shows the assembly file `ex1.asm` with the following code:

```

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

```

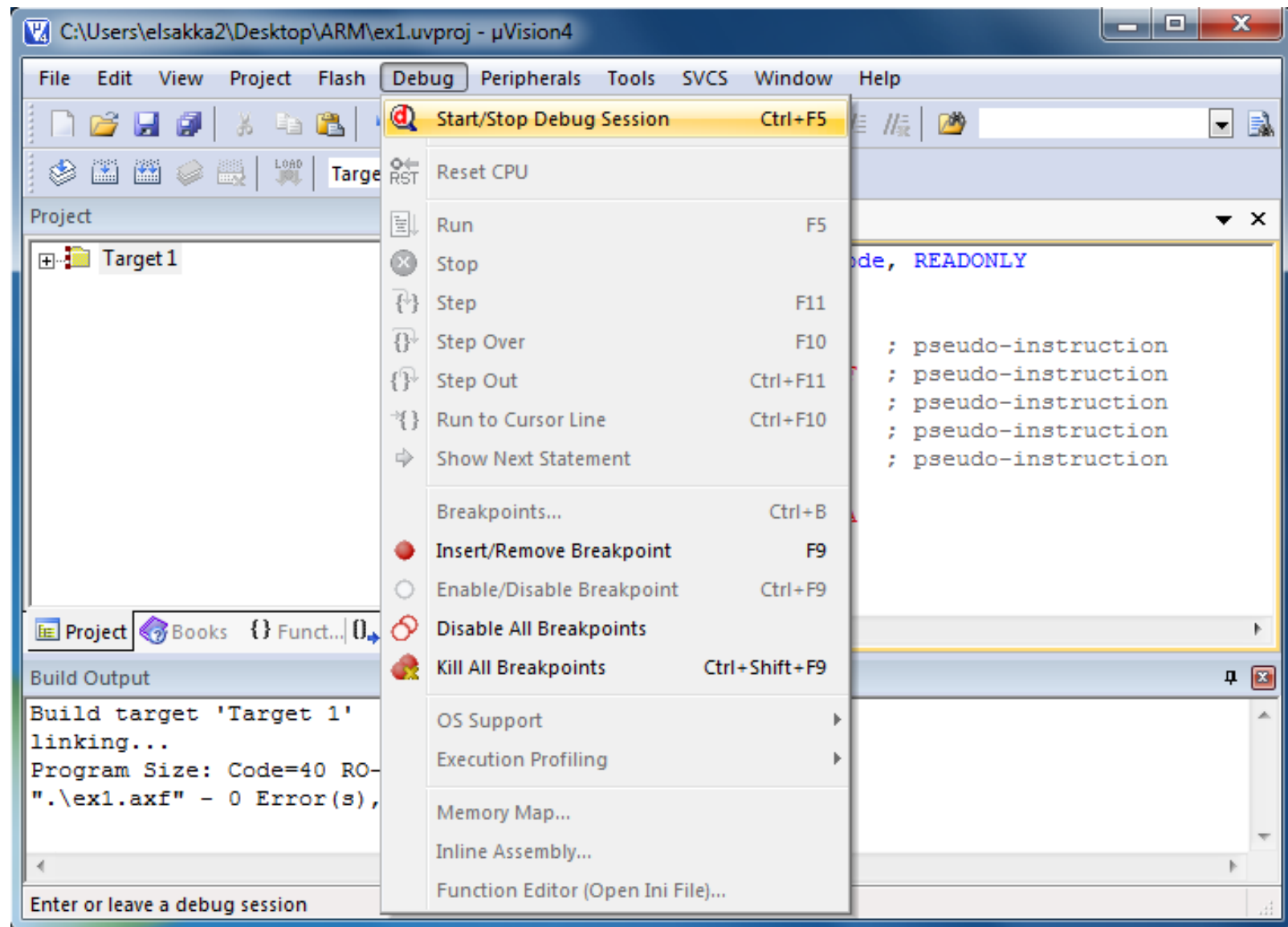
The Build Output window at the bottom shows the following text:

```

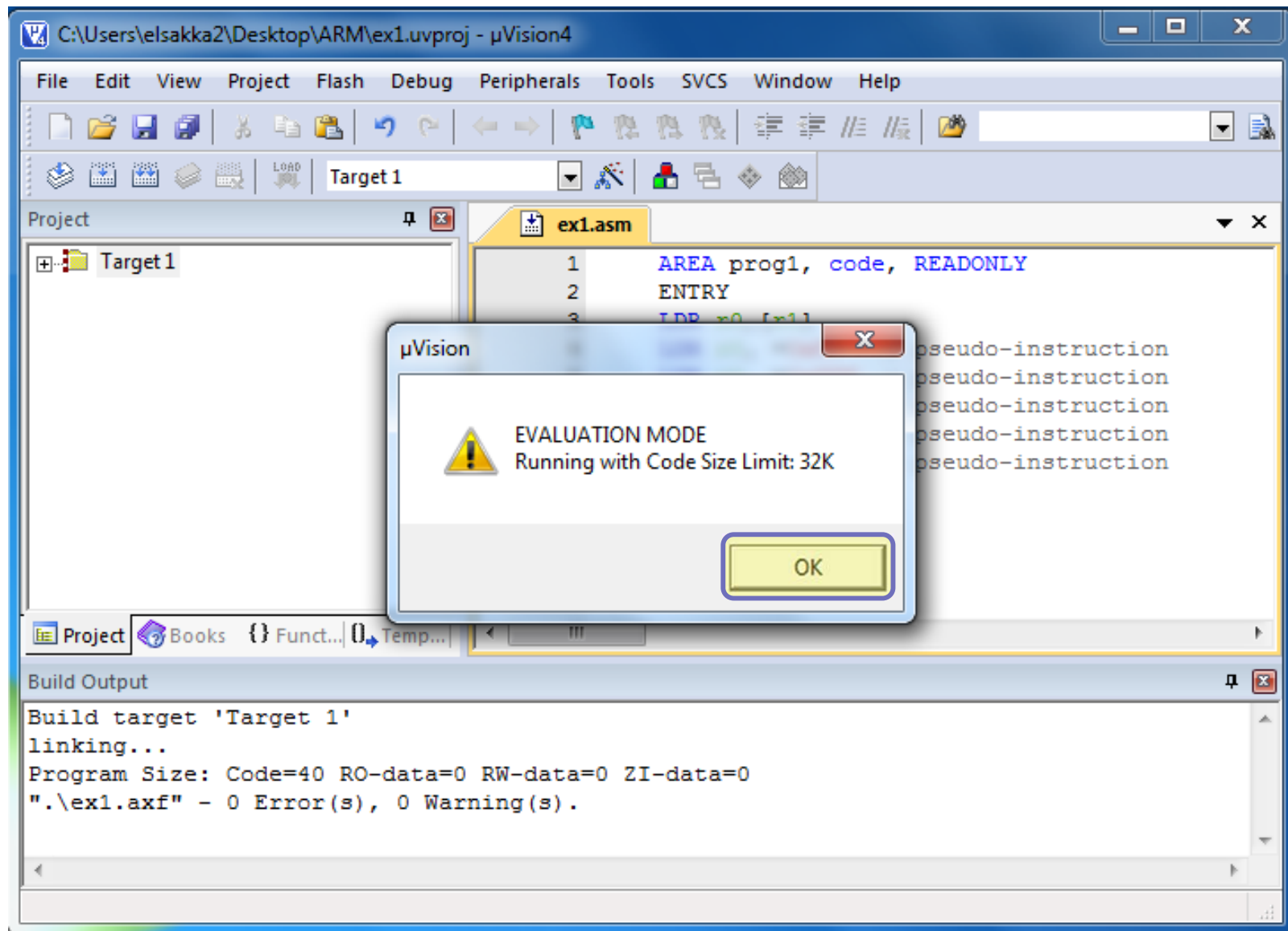
Build target 'Target 1'
linking...
Program Size: Code=40 RO-data=0 RW-data=0 ZI-data=0
".\ex1.axf" - 0 Error(s), 0 Warning(s).

```

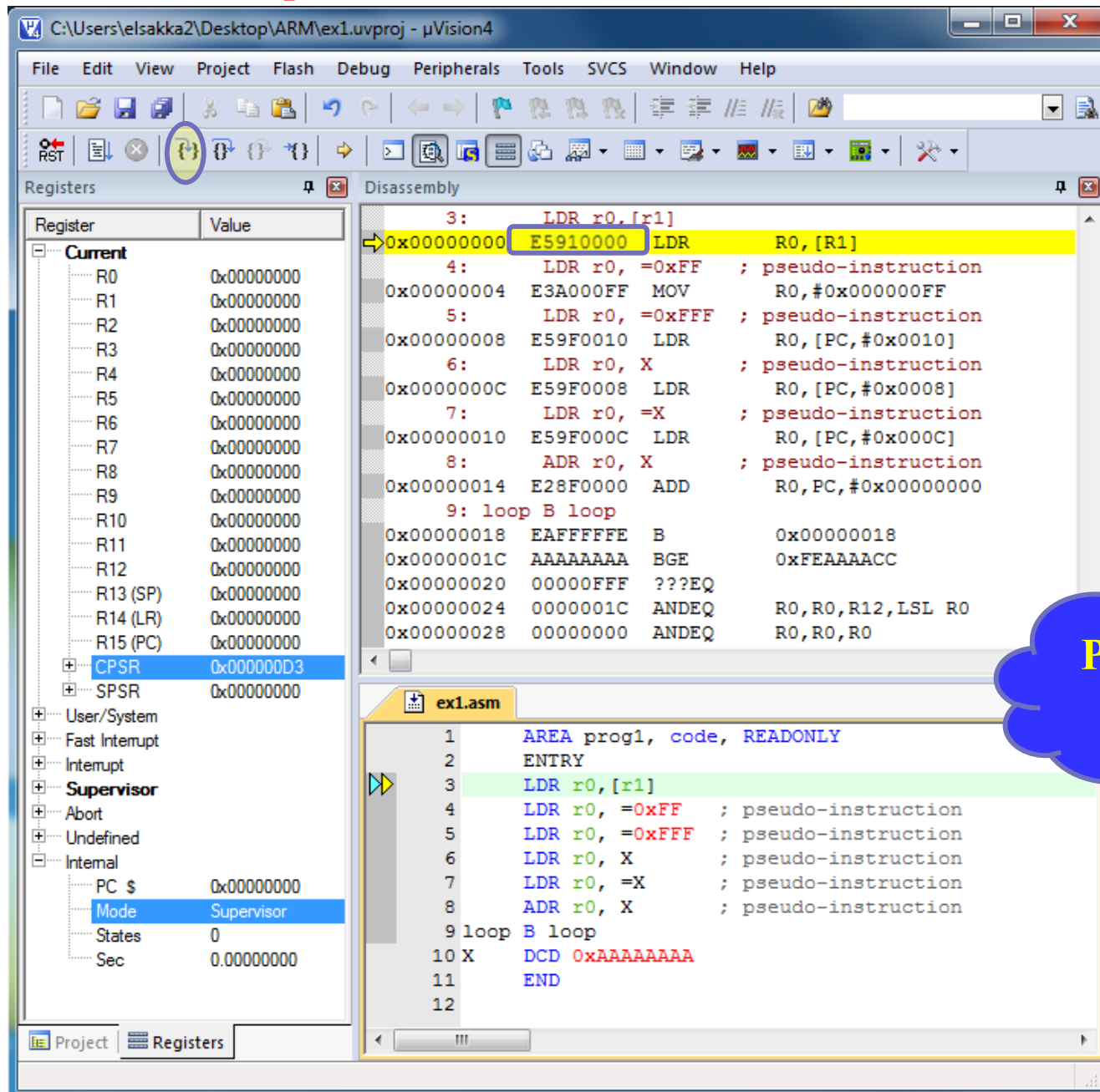
ARM pseudo-instructions



ARM pseudo-instructions



ARM pseudo-instructions



Press Step,
or F11

ARM pseudo-instructions

The screenshot displays the uVision4 IDE interface. The **Registers** window on the left shows the current state of ARM registers. R0 contains 0xE5910000, and R15 (PC) contains 0x00000004. The **Disassembly** window shows the instruction stream, with instruction 4 (LDR r0, =0xFF) highlighted in yellow. The **Source** window at the bottom shows the assembly code for ex1.asm, with instruction 4 (LDR r0, =0xFF) highlighted in green. A blue cloud callout points to the Step button in the toolbar, indicating the next action.

Register	Value
R0	0xE5910000
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

```

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     0xFEAAAAACC
0x00000020 0000FFFF ???EQ
0x00000024 0000001C ANDEQ   R0,R0,R12,LSL R0
0x00000028 00000000 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 0xAAAAAAA
11 END
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 following components:

- Registers Window:**

Register	Value
R0	0x000000FF
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)	0x00000008
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,=0xFFFF ; 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     0xFEAAAAAC
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,=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
12     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:**

Register	Value
R0	0x000000FF
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)	0x0000000C
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,=0xFFFF ; 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 0000FFF ???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,=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
12

```

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 Disassembly window open. The PC register (R15) is highlighted in the registers list on the left, showing a value of 0x00000010. The Disassembly window shows the following instructions:

Address	Instruction	Comment
0x00000000	E5910000 LDR R0, [R1]	
0x00000004	E3A000FF MOV R0, #0x000000FF	; pseudo-instruction
0x00000008	E59F0010 LDR R0, [PC, #0x0010]	
0x0000000C	E59F0008 LDR R0, [PC, #0x0008]	
0x00000010	E59F000C LDR R0, [PC, #0x000C]	
0x00000014	E28F0000 ADD R0, PC, #0x00000000	
0x00000018	EAF FFFFE B 0x00000018	
0x0000001C	AAAAAAA BGE 0xFEAAAAACC	
0x00000020	0000FFF ???EQ	
0x00000024	0000001C ANDEQ R0, R0, R12, LSL R0	
0x00000028	00000000 ANDEQ R0, R0, R0	

The assembly source file (ex1.asm) is also visible at the bottom:

```

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
12
  
```

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

The screenshot displays the uVision4 IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, and Window. The main workspace is divided into three panes:

- Registers:** A table showing the current state of ARM registers. R15 (PC) is highlighted with a value of 0x00000014.
- Disassembly:** A list of instructions with their memory addresses and machine codes. Instruction 8 at address 0x00000014 is highlighted, showing the pseudo-instruction 'ADR r0, X' translated to 'E28F0000 ADD R0, PC, #0x00000000'.
- Source:** The assembly code for 'ex1.asm'. Instruction 8 is highlighted, showing 'ADR r0, X ; pseudo-instruction'.

A blue arrow points from the text bubble on the left to the PC register in the Registers pane. Another blue arrow points from the text bubble on the right to the highlighted instruction in the Disassembly pane.

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 branch instruction) highlighted in yellow. The **Source Code** window at the bottom shows the assembly code for 'ex1.asm', with the corresponding 'loop B loop' instruction highlighted in green.

Address	Instruction	Comment
0x00000000	E5910000 LDR R0, [R1]	
0x00000004	E3A000FF MOV R0, #0x000000FF	; pseudo-instruction
0x00000008	E59F0010 LDR R0, [PC, #0x0010]	
0x0000000C	E59F0008 LDR R0, [PC, #0x0008]	
0x00000010	E59F000C LDR R0, [PC, #0x000C]	
0x00000014	E28F0000 ADD R0, PC, #0x00000000	
0x00000018	EAF00000 B 0x00000018	; pseudo-instruction
0x0000001C	AAAAAA BGE 0xFEAAAA	
0x00000020	0000FFF ???EQ	
0x00000024	0000001C ANDEQ R0, R0, R12, LSL R0	
0x00000028	00000000 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

Disassembly

```

3: 0x00000000 E5910000 LDR R0,[R1]
4: 0x00000004 E3A000FF MOV R0,#0x000000FF ; pseudo-instruction
5: 0x00000008 E59F0010 LDR R0,[PC,#0x0010] ; pseudo-instruction
6: 0x0000000C E59F0008 LDR R0,[PC,#0x0008] ; pseudo-instruction
7: 0x00000010 E59F000C LDR R0,[PC,#0x000C] ; pseudo-instruction
8: 0x00000014 E28F0000 ADD R0,PC,#0x00000000 ; pseudo-instruction
9: loop B loop
10: 0x00000018 EAffFFFF B 0x00000018
11: 0x0000001C AAAAAAAA BGE 0xFEAAAAAC
12: 0x00000020 0000FFFF ???EQ
13: 0x00000024 0000001C ANDEQ R0,R0,R12,LSL R0
14: 0x00000028 00000000 ANDEQ R0,R0,R0

```

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
12

```

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)	0x00000020
R14 (LR)	0x00000000
R15 (PC)	0x00000018

Disassembly

```

3: 0x00000000 E5910000 LDR R0,[R1]
4: 0x00000004 E3A000FF MOV R0,#0x000000FF ; pseudo-instruction
5: 0x00000008 E59F000C LDR R0,[PC,#0x000C] ; pseudo-instruction
6: 0x0000000C E59F0010 LDR R0,[PC,#0x0010] ; pseudo-instruction
7: 0x00000010 E59F0008 LDR R0,[PC,#0x0008] ; pseudo-instruction
8: 0x00000014 E28F0000 ADD R0,PC,#0x00000000 ; pseudo-instruction
9: loop B loop
10: 0x00000018 EAffFFFF B 0x00000018
11: 0x0000001C 0000FFFF ???EQ
12: 0x00000020 00000024 ANDEQ R0,R0,R4,LSR #32
13: 0x00000024 AAAAAAAA BGE 0xFEAAAAD4
14: 0x00000028 00000000 ANDEQ R0,R0,R0

```

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
14

```