

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:

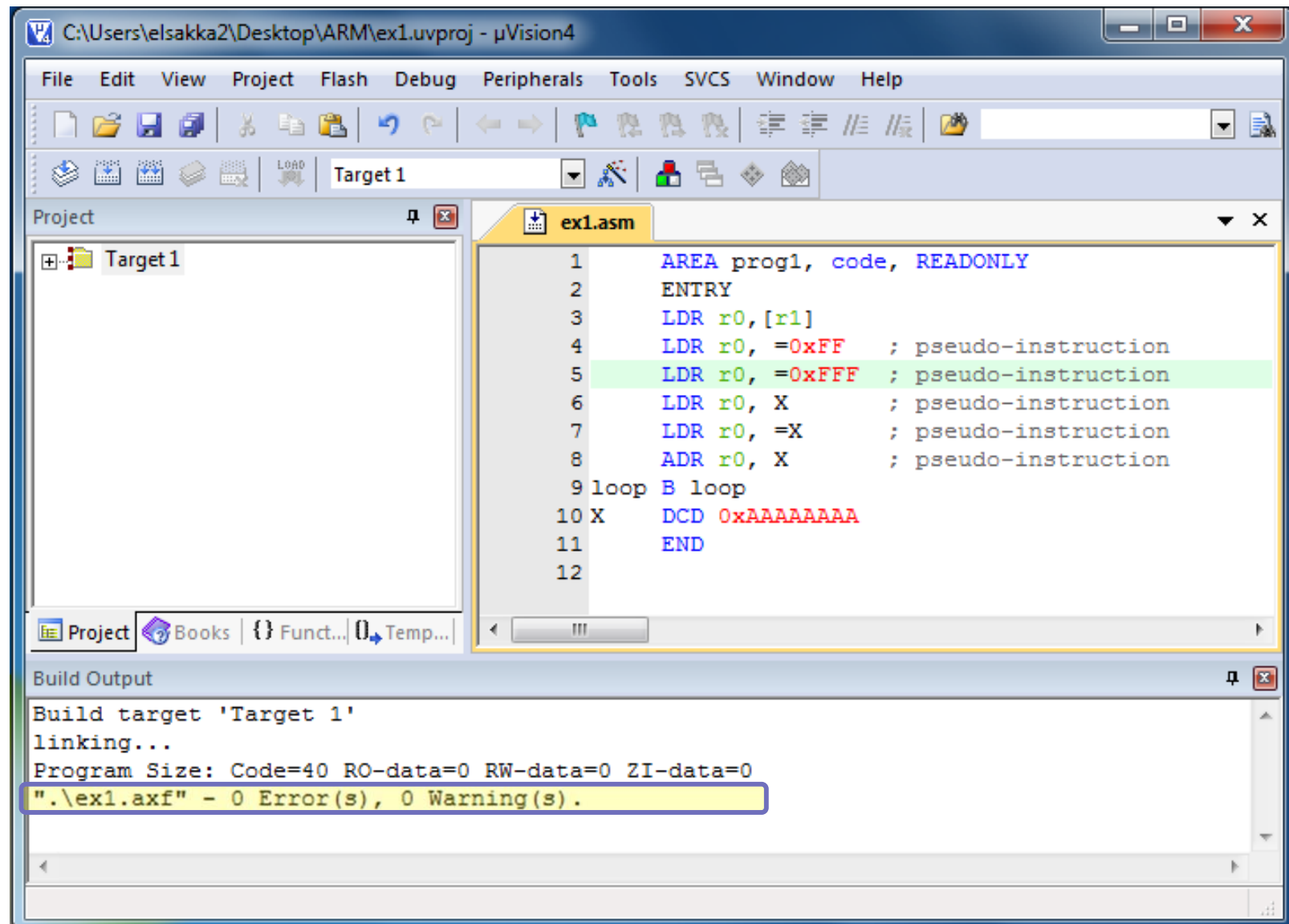
```

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

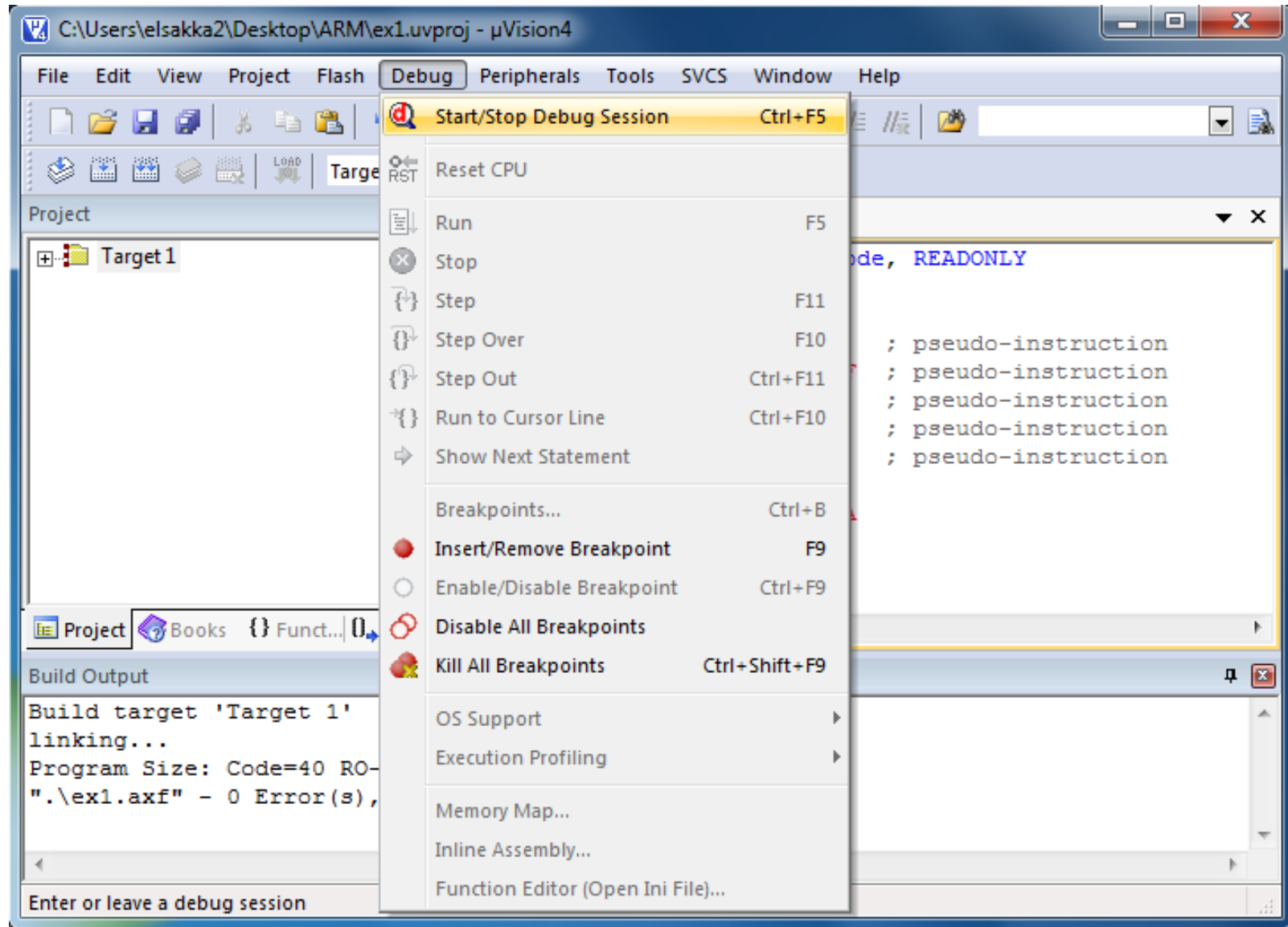
loop B loop
X    DCD 0xAAAAAAAA
END

```

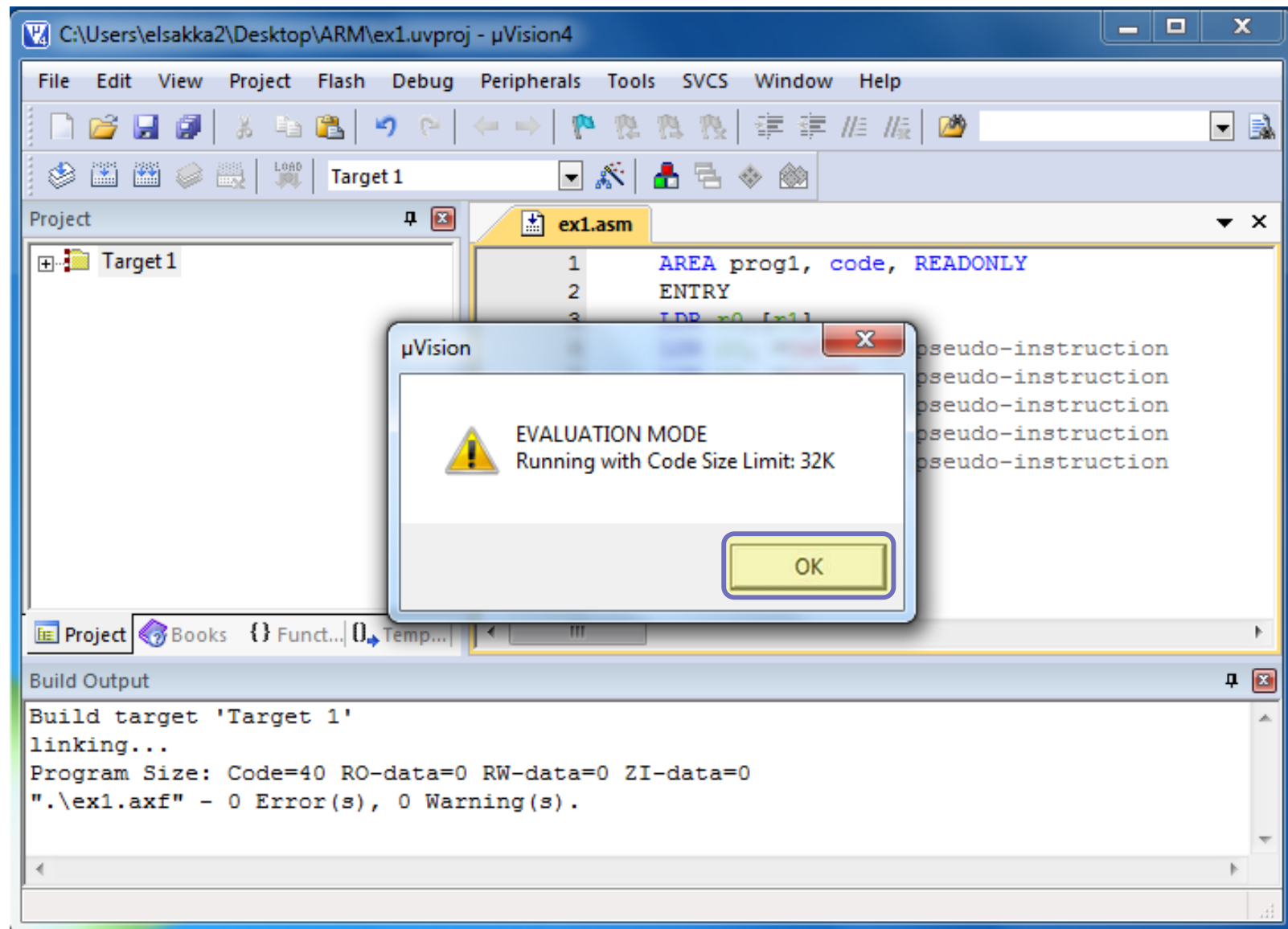
ARM pseudo-instructions



ARM pseudo-instructions



ARM pseudo-instructions



ARM pseudo-instructions

The screenshot displays the uVision4 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		
0x00000018	EAF00000	B	0x00000018
0x0000001C	AAAA0000	BGE	0xFEAAAA00
0x00000020	00000000	???	EQ
0x00000024	0000001C	ANDEQ	R0, R0, R12, LSL R0
0x00000028	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 0xAAAAAAAA
11  END
12

```

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. Instruction 4, `LDR r0, =0xFF`, is highlighted in yellow and labeled as a pseudo-instruction. A blue callout bubble with the text "Press Step, or F11" points to the Step button in the toolbar.

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 0xAAAAAAAA
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     0xFEAAAACC
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 END
12

```


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, =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, 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 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

Registers

Register	Value
R0	0xAAAAAAAA
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)	0x00000010
CPSR	0x000000D3
SPSR	0x00000000

Disassembly

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]	; pseudo-instruction
0x00000014	E28F0000 ADD R0, PC, #0x00000000	; pseudo-instruction
0x00000018	EAF00000 B 0x00000018	
0x0000001C	AAAA0000 BGE 0xFEAAAA00	
0x00000020	000000FF ???EQ	
0x00000024	0000001C ANDEQ R0, R0, R12, LSL R0	
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 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 shows the uVision4 IDE with the following components:

- Registers Window:** Shows the current state of ARM registers. R15 (PC) is highlighted with a value of 0x00000014.
- Disassembly Window:** Shows the instruction at location 0x00000014: `ADR r0, X`. The disassembler has translated this to `E28F0000 ADD R0, PC, #0x00000000`. The instruction is highlighted in yellow.
- Source Window:** Shows the assembly code for 'ex1.asm'. The instruction `ADR r0, X` is highlighted in green.

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. R15 (PC) is highlighted with a value of 0x00000018. The **Disassembly** window shows the instruction stream, with instruction 9 (loop B loop) highlighted. The **ex1.asm** window at the bottom shows the source code for the program, including pseudo-instructions like `LDR r0, =0xFF` and `LDR r0, =0xFFFF`.

Address	Instruction	Comment
0x00000000	LDR r0, [r1]	
0x00000004	LDR r0, =0xFF	; pseudo-instruction
0x00000008	LDR r0, =0xFFFF	; pseudo-instruction
0x0000000C	LDR r0, X	; pseudo-instruction
0x00000010	LDR r0, =X	; pseudo-instruction
0x00000014	ADR r0, X	; pseudo-instruction
0x00000018	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, =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 0xAAAAAAA
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: EAFEEEE B 0x00000018
11: AAAAAA BGE 0xFEAAAAAC
12: 00000020 000000FF ???EQ
13: 00000024 0000001C ANDEQ R0, R0, R12, LSL R0
14: 00000028 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 0xAAAAAAA
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)	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: EAFEEEE B 0x00000018
11: 0000001C 000000FF ???EQ
12: 00000020 00000024 ANDEQ R0, R0, R4, LSR #32
13: AAAAAA BGE 0xFEAAAAD4
14: 00000028 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 0xAAAAAAA
13 END
14

```