Pontificia Universidad
JAVERIANA
Cali

ACREDITACIÓN INSTITUCIONAL DE ALTA CALIDAD
POR 8 AÑOS

Res. 2333 del 2012

Vigilada Mineducación Resolución 12220 de 2016

ARQUITECTURA DE COMPUTADORES II: LABORATORY.

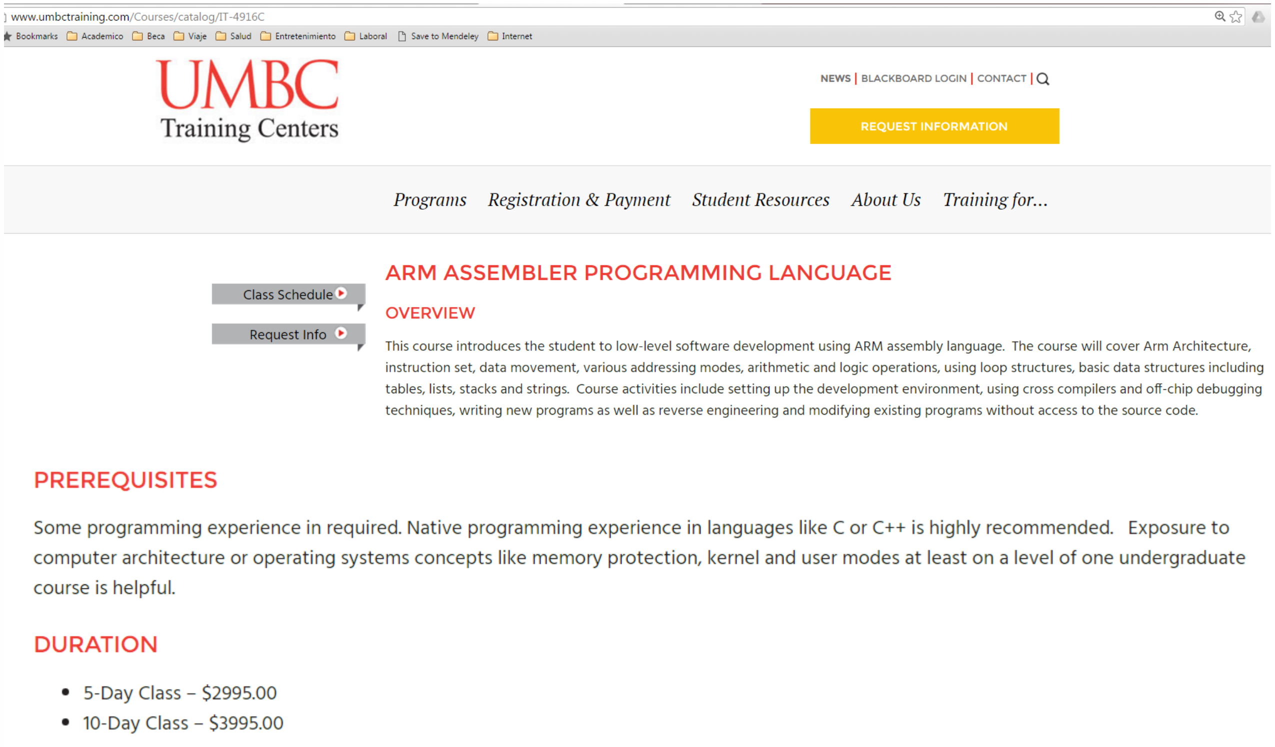
Professor: Roger Gomez Nieto, MSc

roger.gomez@javerianacali.edu.co

Subject: AC182_

Session 1.

- IDE’s installation.
- Presentation and mail list.
- Score, bibliography and course overview.
- Previous Knowledge.
- Introduction to ASM ARM.
- “Hello world” in ASM.



ARM ASSEMBLER PROGRAMMING LANGUAGE

OVERVIEW

This course introduces the student to low-level software development using ARM assembly language. The course will cover Arm Architecture, instruction set, data movement, various addressing modes, arithmetic and logic operations, using loop structures, basic data structures including tables, lists, stacks and strings. Course activities include setting up the development environment, using cross compilers and off-chip debugging techniques, writing new programs as well as reverse engineering and modifying existing programs without access to the source code.

PREREQUISITES

Some programming experience in required. Native programming experience in languages like C or C++ is highly recommended. Exposure to computer architecture or operating systems concepts like memory protection, kernel and user modes at least on a level of one undergraduate course is helpful.

DURATION

- 5-Day Class – $2995.00
- 10-Day Class – $3995.00

Pontificia Universidad
JAVERIANA
Cali

ACREDITACIÓN INSTITUCIONAL DE ALTA CALIDAD
POR 8 AÑOS

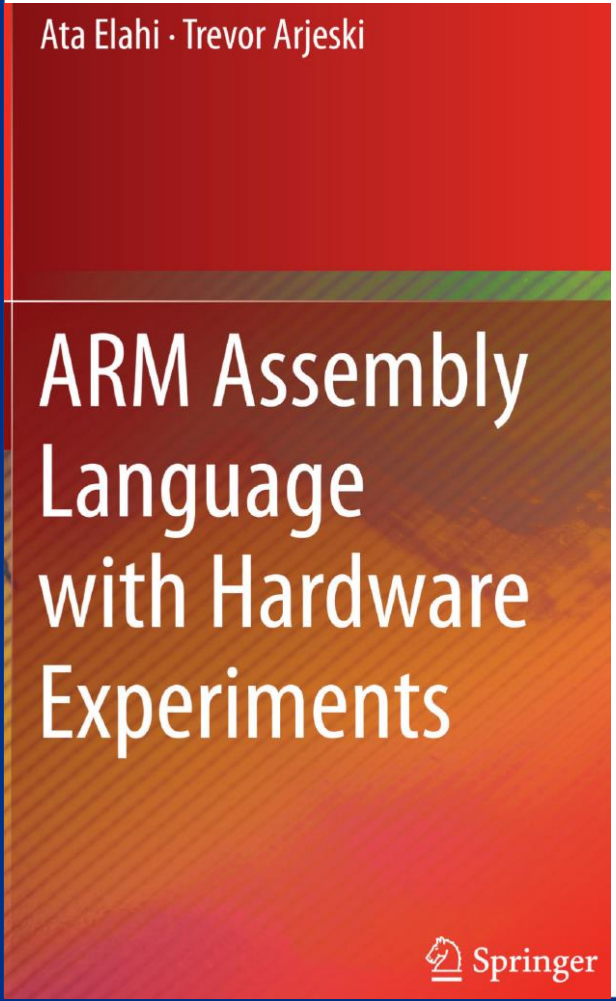
TOPICS

- Arm history and ecosystem
- ARM Architecture
- Instruction Set
- Addressing Modes
- Programs
- Data Movement
- Logic
- Arithmetic
- Barrel shifter
- Conditional execution
- Program Loops
- Strings
- Tables and Lists
- Stacks
- Subroutines
- Interrupts
- Code Conversion
- Cross compiling
- Reverse engineering

Course Evaluation.

• Laboratory:	10%.
• Final Project:	25%.

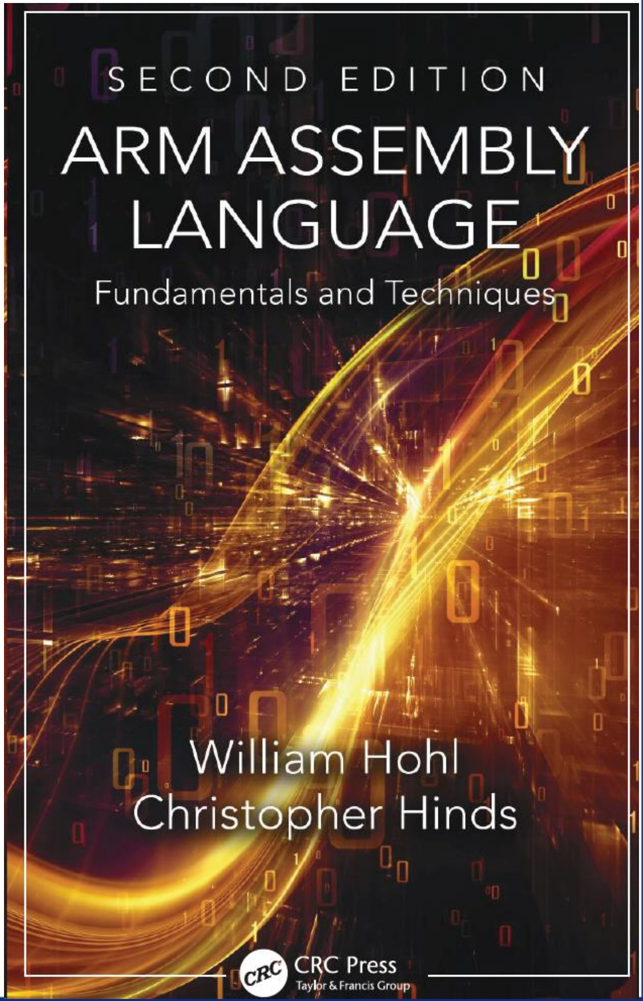
Bibliografía



Ata Elahi · Trevor Arjeski

ARM Assembly Language with Hardware Experiments

Springer



SECOND EDITION

ARM ASSEMBLY LANGUAGE

Fundamentals and Techniques

William Hohl
Christopher Hinds

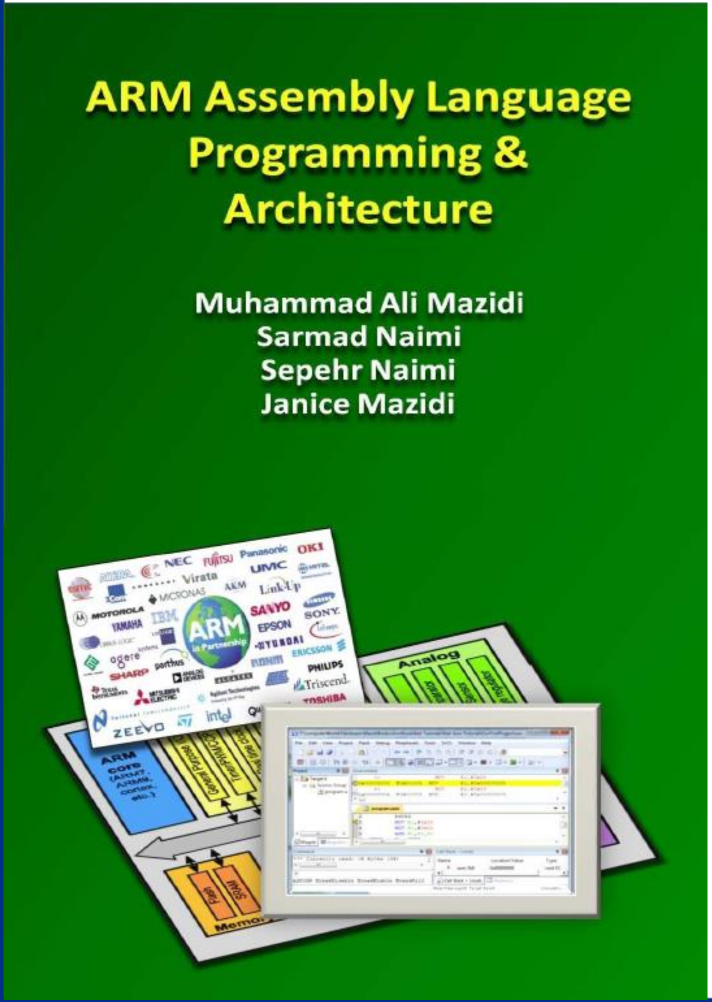
CRC Press
Taylor & Francis Group

Bibliografía

Pontificia Universidad
JAVERIANA
Cali

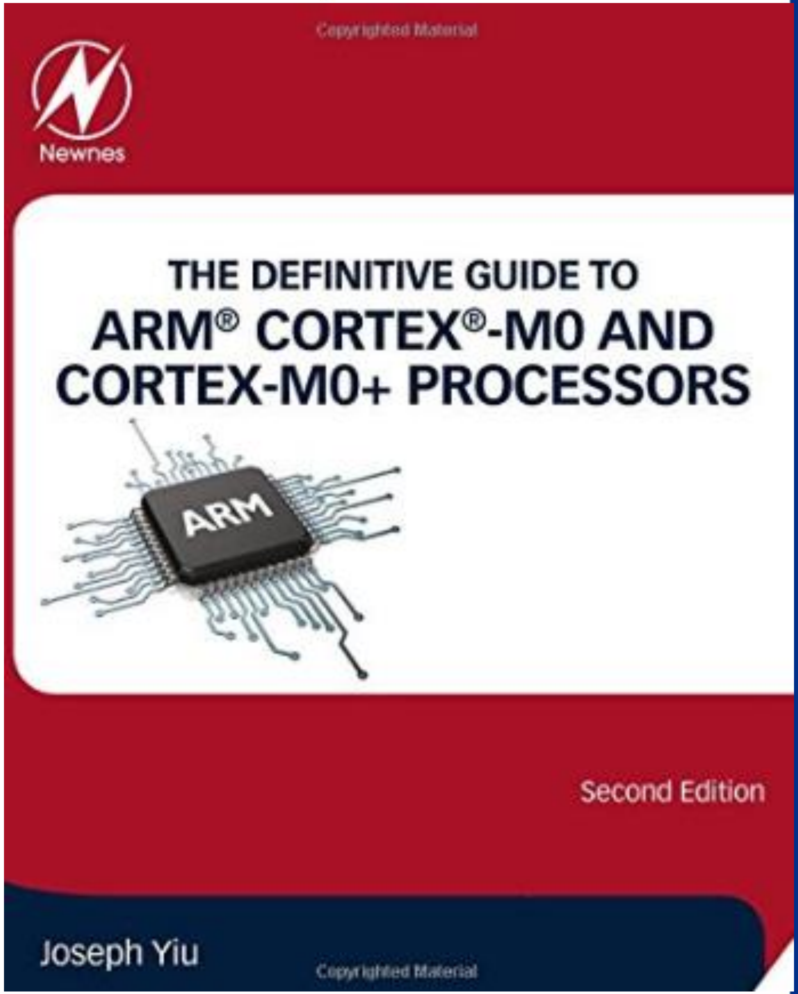
Res. 2333 del 2012

Vigilada Mineducación Resolución 12220 de 2016



ARM Assembly Language Programming & Architecture

Muhammad Ali Mazidi
Sarmad Naimi
Sepehr Naimi
Janice Mazidi



Copyrighted Material

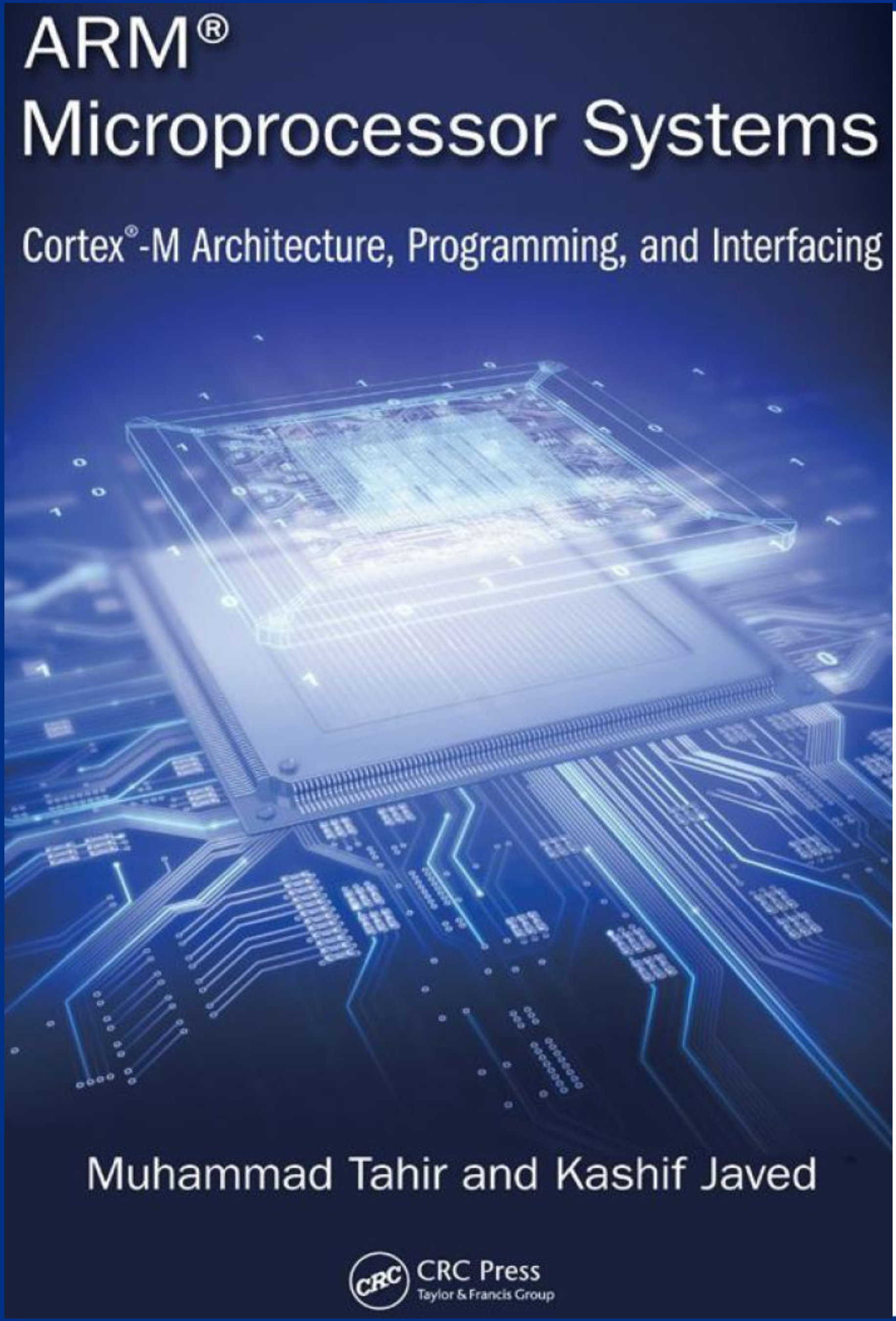
Newnes

THE DEFINITIVE GUIDE TO
ARM® CORTEX®-M0 AND
CORTEX-M0+ PROCESSORS

Second Edition

Joseph Yiu

Copyrighted Material



ARM®
Microprocessor Systems

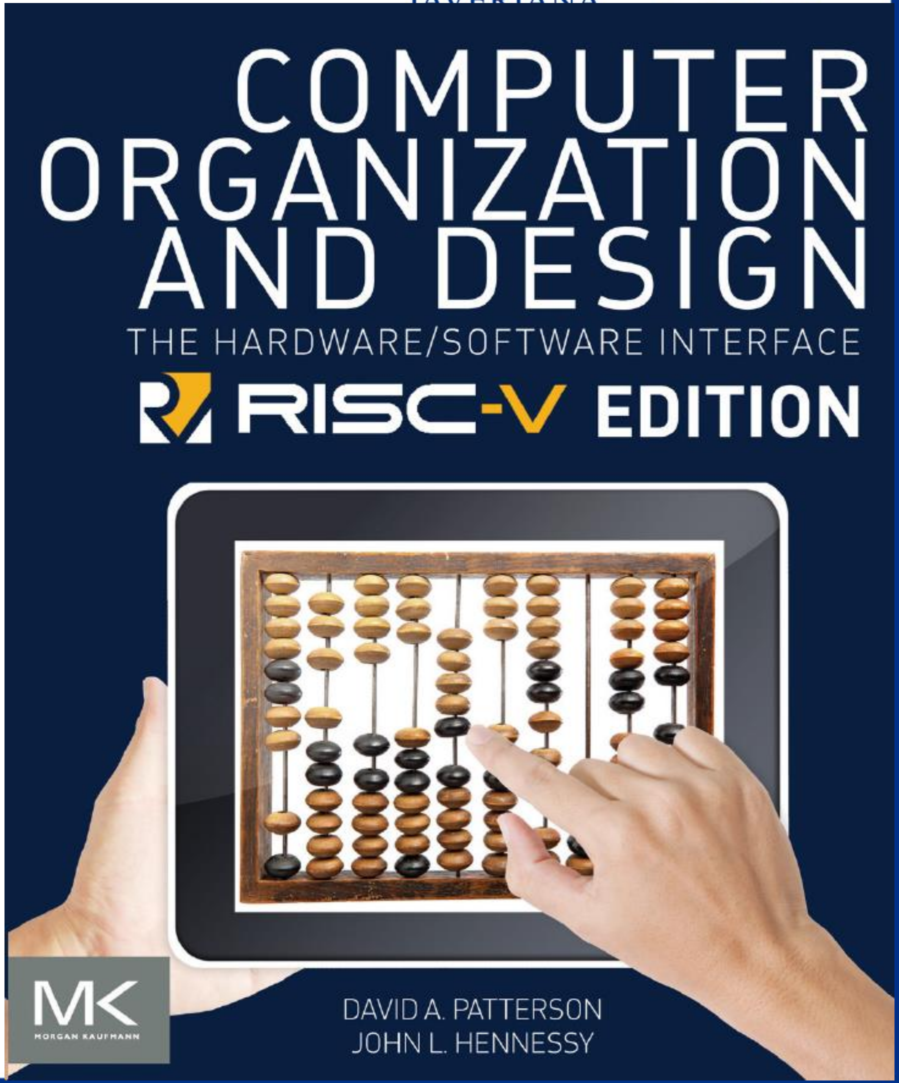
Cortex®-M Architecture, Programming, and Interfacing

Muhammad Tahir and Kashif Javed

CRC Press
Taylor & Francis Group

Pontificia Universidad
JAVERIANA

ACREDITACIÓN INSTITUCIONAL DE ALTA CALIDAD
POR 8 AÑOS



COMPUTER ORGANIZATION AND DESIGN

THE HARDWARE/SOFTWARE INTERFACE

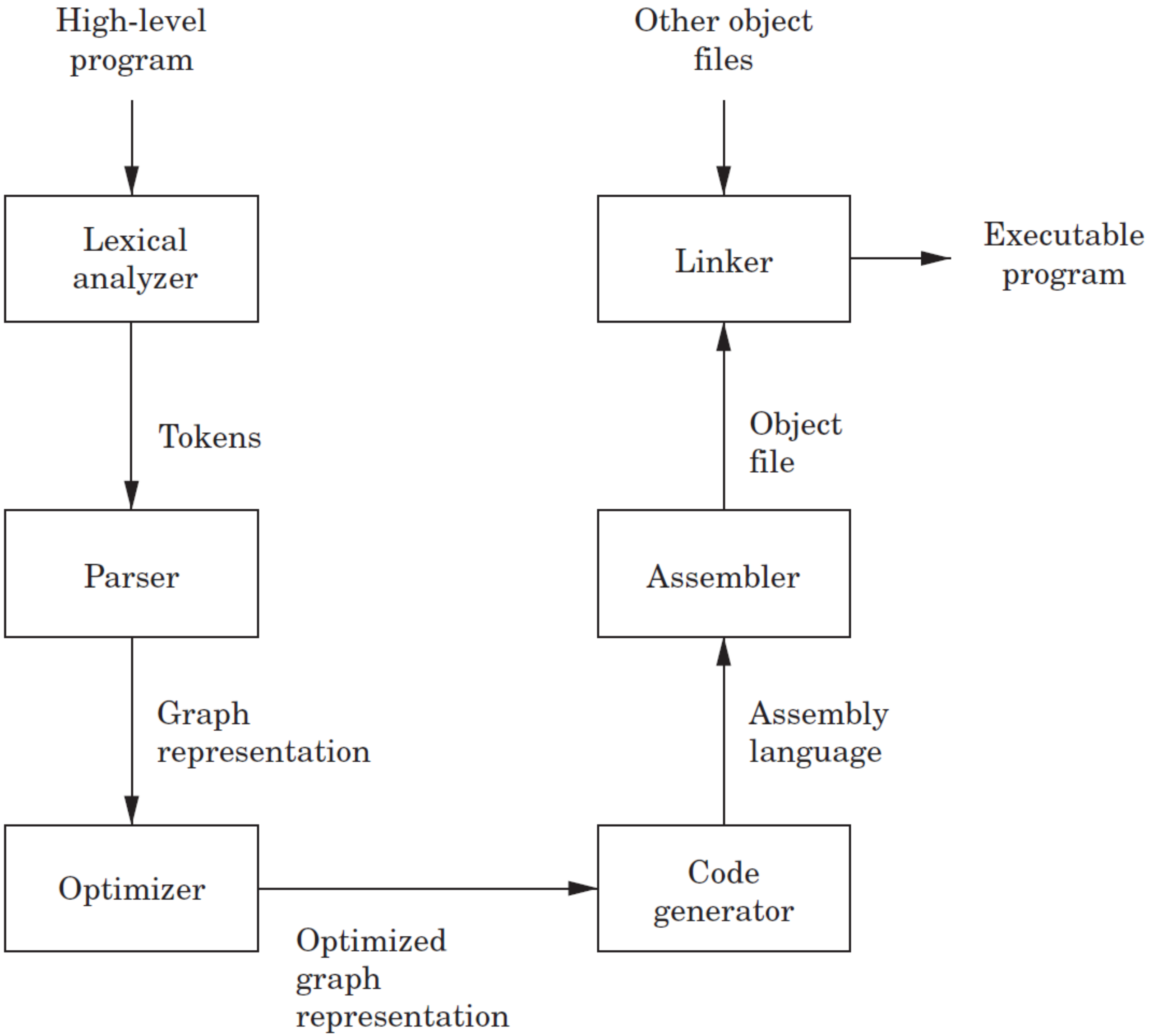
RISC-V EDITION

MK
MORGAN KAUFMANN

DAVID A. PATTERSON
JOHN L. HENNESSY

Reasons to Learn Assembly

- the first steps in booting the computer,
- code to handle interrupts,
- low-level locking code for multi-threaded programs,
- code for machines where no compiler exists,
- code which needs to be optimized beyond the limits of the compiler,
- on computers with very limited memory, and
- code that requires low-level access to architectural and/or processor features.



High-level program

Lexical analyzer

Tokens

Parser

Graph representation

Optimizer

Optimized graph representation

Code generator

Assembly language

Assembler

Object file

Other object files

Linker

Executable program

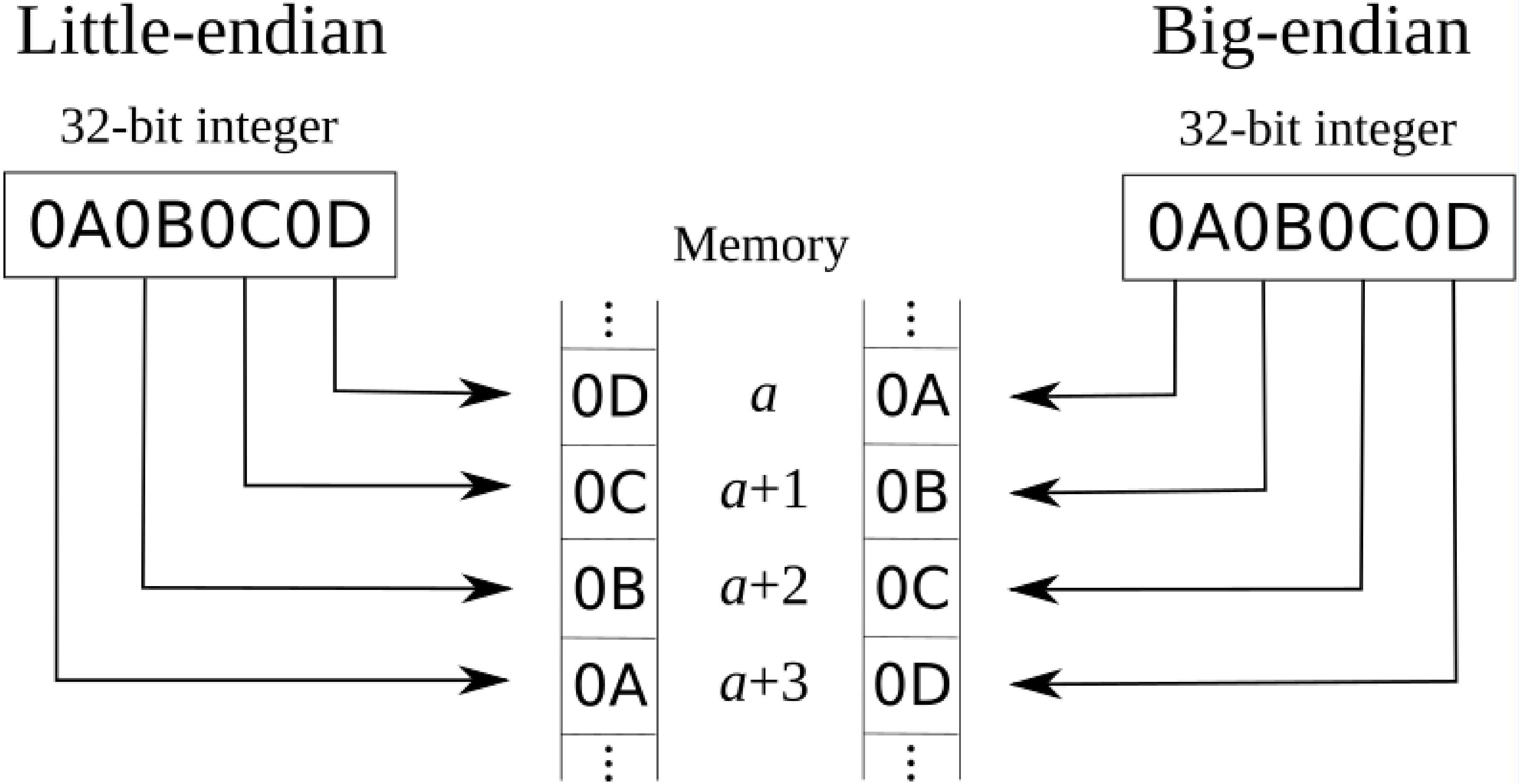
Figure 1.2
Stages of a typical compilation sequence.

Little Endian

Pontificia Universidad JAVERIANA Cali

Res. 2333 del 2012

Vigilada Mineducación Resolución 12220 de 2016

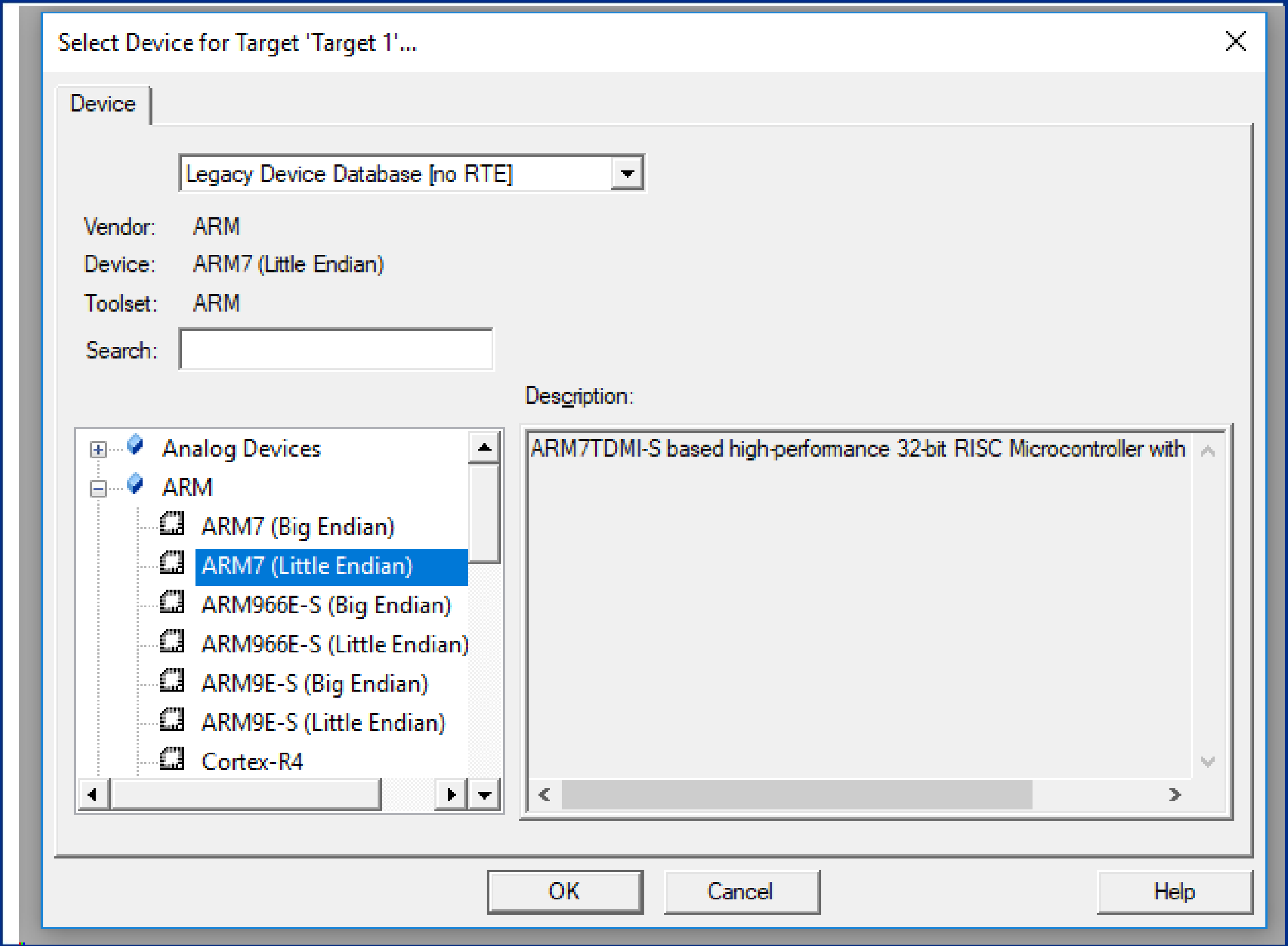


IDE.

- https://www.keil.com/download/product/ versión 5.25 MDK (Microcontroller Development Kit) (837 MB).

- Fill the required information.

- http://www2.keil.com/mdk5/legacy/ Legacy Support for ARM7 (127 MB).



Select Device for Target 'Target 1'...

Device

Legacy Device Database [no RTE]

Vendor: ARM

Device: ARM7 (Little Endian)

Toolset: ARM

Search:

Description:

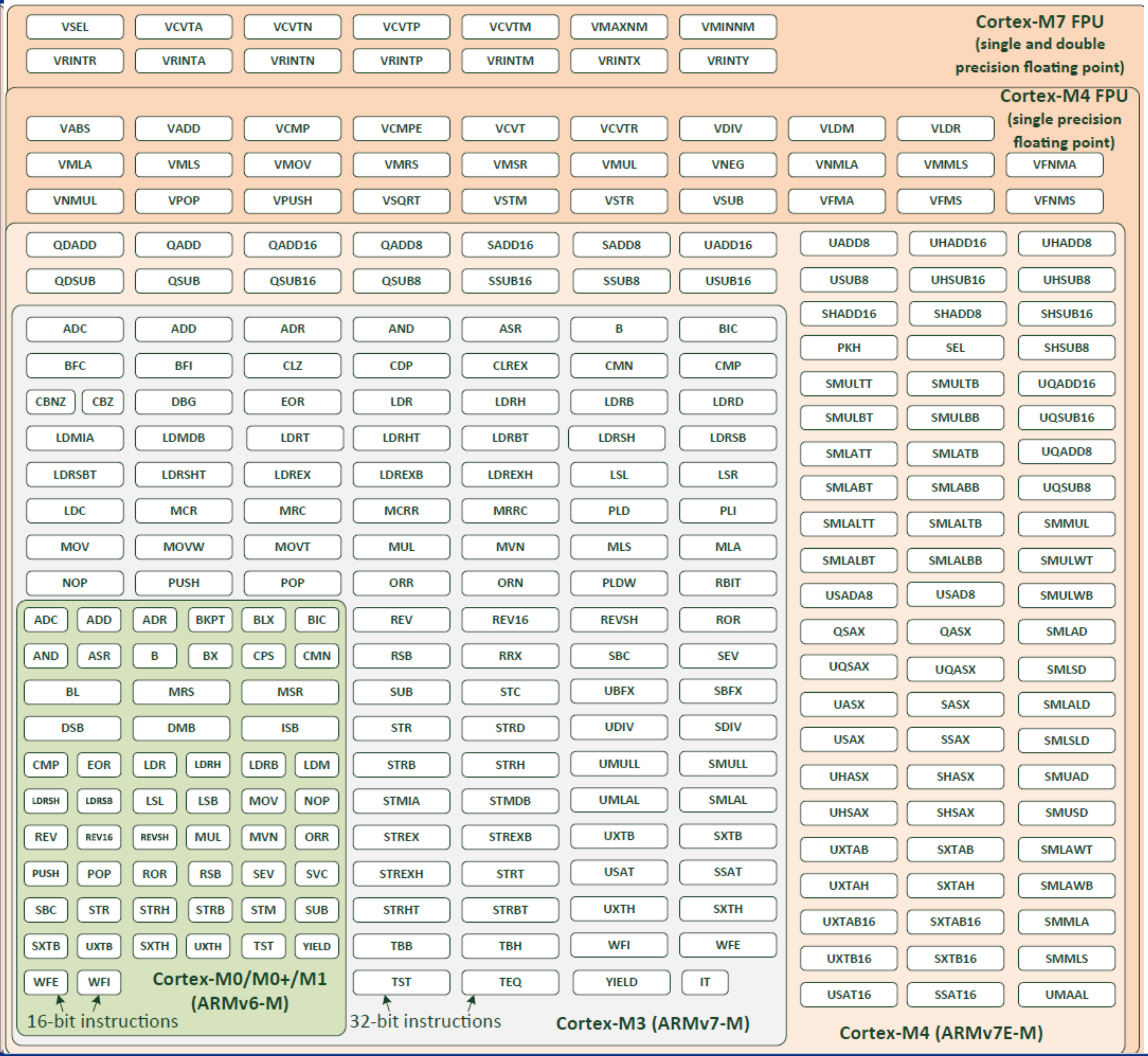
ARM7TDMI-S based high-performance 32-bit RISC Microcontroller with

Instruction Set Cortex -M family

Pontificia Universidad
JAVERIANA
Cali

Res. 2333 del 2012

Vigilada Mineducación Resolución 12220 de 2016



Cortex-M7 FPU
(single and double precision floating point)

VSEL	VCVTA	VCVTN	VCVTP	VCVTM	VMAXNM	VMINNM
VRINTR	VRINTA	VRINTN	VRINTP	VRINTM	VRINTX	VRINTY

Cortex-M4 FPU
(single precision floating point)

VABS	VADD	VCMP	VCMPE	VCVT	VCVTR	VDIV	VLDM	VLDR	
VMLA	VMLS	VMOV	VMRS	VMSR	VMUL	VNEG	VNMLA	VMMLS	VFNMA
VNMUL	VPOP	VPUSH	VSQRT	VSTM	VSTR	VSUB	VFMA	VFMS	VFNMS

Cortex-M4 (ARMv7E-M)

QDADD	QADD	QADD16	QADD8	SADD16	SADD8	UADD16	UADD8	UHADD16	UHADD8
QDSUB	QSUB	QSUB16	QSUB8	SSUB16	SSUB8	USUB16	USUB8	UHSUB16	UHSUB8

SHADD16	SHADD8	SHSUB16
PKH	SEL	SHSUB8
SMULTT	SMULTB	UQADD16
SMULBT	SMULBB	UQSUB16
SMLATT	SMLATB	UQADD8
SMLABT	SMLABB	UQSUB8
SMLALTT	SMLALTB	SMMUL
SMLALBT	SMLALBB	SMULWT
USADA8	USAD8	SMULWB
QSAX	QASX	SMLAD
UQSAX	UQASX	SMLSD
UASX	SASX	SMLALD
USAX	SSAX	SMLSLD
UHASX	SHASX	SMUAD
UHSAX	SHSAX	SMUSD
UXTAB	SXTAB	SMLAWT
UXTAH	SXTAH	SMLAWB
UXTAB16	SXTAB16	SMMLA
UXTB16	SXTB16	SMMLS
USAT16	SSAT16	UMAAL

Cortex-M3 (ARMv7-M)

ADC	ADD	ADR	AND	ASR	B	BIC
BFC	BFI	CLZ	CDP	CLREX	CMN	CMP
CBNZ CBZ	DBG	EOR	LDR	LDRH	LDRB	LDRD
LDMIA	LDMDB	LDRT	LDRHT	LDRBT	LDRSH	LDRSB
LDRSBT	LDRSHT	LDREX	LDREXB	LDREXH	LSL	LSR
LDC	MCR	MRC	MCRR	MRRC	PLD	PLI
MOV	MOVW	MOVT	MUL	MVN	MLS	MLA
NOP	PUSH	POP	ORR	ORN	PLDW	RBIT

REV	REV16	REVSH	ROR
RSB	RRX	SBC	SEV
SUB	STC	UBFX	SBFX
STR	STRD	UDIV	SDIV
STRB	STRH	UMULL	SMULL
STMIA	STMDB	UMLAL	SMLAL
STREX	STREXB	UXTB	SXTB
STREXH	STRT	USAT	SSAT
STRHT	STRBT	UXTH	SXTH
TBB	TBH	WFI	WFE
TST	TEQ	YIELD	IT

32-bit instructions

Cortex-M0/M0+/M1
(ARMv6-M)

ADC	ADD	ADR	BKPT	BLX	BIC
AND	ASR	B	BX	CPS	CMN
BL		MRS		MSR	
DSB		DMB		ISB	
CMP	EOR	LDR	LDRH	LDRB	LDM
LDRSH	LDRSB	LSL	LSB	MOV	NOP
REV	REV16	REVSH	MUL	MVN	ORR
PUSH	POP	ROR	RSB	SEV	SVC
SBC	STR	STRH	STRB	STM	SUB
SXTB	UXTB	SXTH	UXTH	TST	YIELD
WFE	WFI				

16-bit instructions

ARM Directives

Directive	Description
AREA	Instructs the assembler to assemble a new code or data section
END	Informs the assembler that it has reached the end of a source file.
ENTRY	Declares an entry point to a program.
EQU	Gives a symbolic name to a numeric constant, a register-relative value or a PC-relative value.
INCLUDE	It adds the contents of a file to our program.

[label] mnemonic [operands] [;comment]

AREA

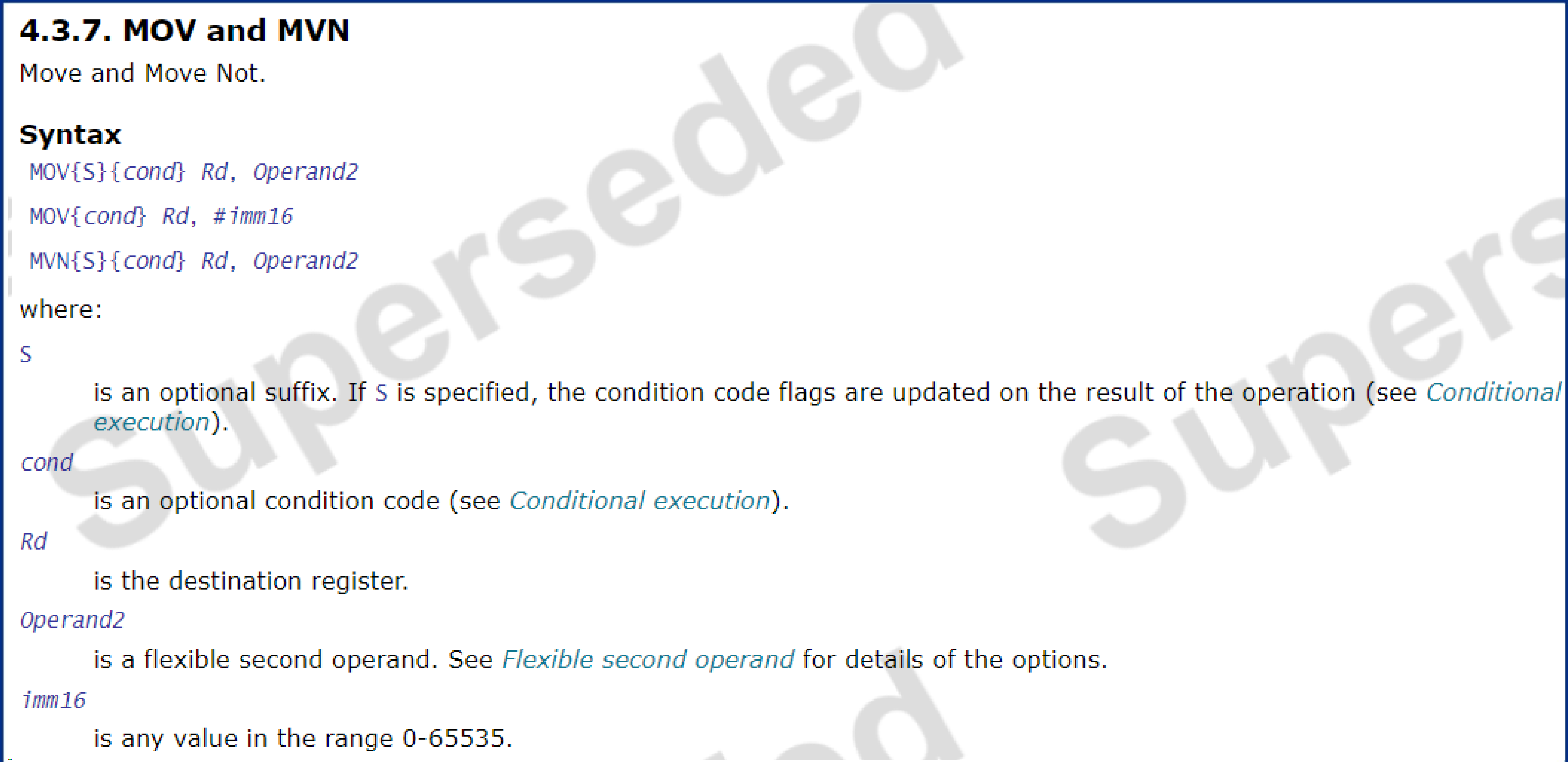
Microprocessor programs often contain several **AREA** statements for the following purposes:

- Reset (startup) address
- Interrupt service addresses
- Trap (software interrupt) addresses
- RAM storage
- Stack
- Main program
- Subroutines
- Input/Output

The AREA statement at the start of an ARM program is required, and its absence will cause the assembly to fail.

END, marks the end of the assembly language source program. This must appear in the file or a "missing END directive" error will occur.

ENTRY is beginning of the code.



4.3.7. MOV and MVN

Move and Move Not.

Syntax

```
MOV{S}{cond} Rd, Operand2
MOV{cond} Rd, #imm16
MVN{S}{cond} Rd, Operand2
```

where:

`S`
is an optional suffix. If `S` is specified, the condition code flags are updated on the result of the operation (see *Conditional execution*).

cond
is an optional condition code (see *Conditional execution*).

Rd
is the destination register.

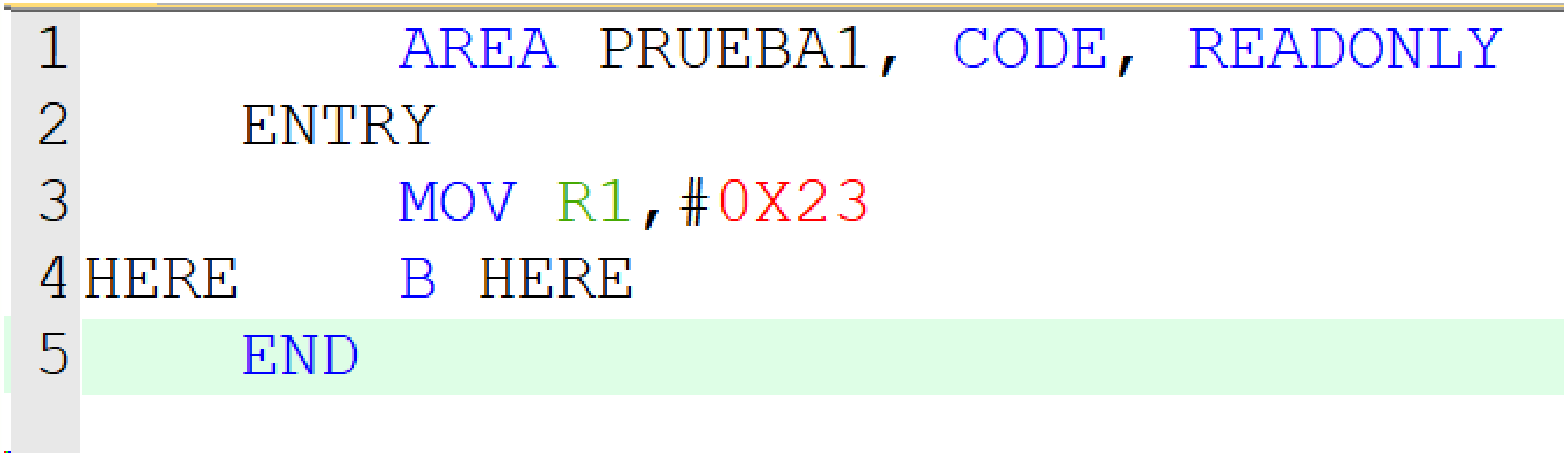
Operand2
is a flexible second operand. See *Flexible second operand* for details of the options.

imm16
is any value in the range 0-65535.

- Notice the # before immediate value.
- “immediate”” is a constant value that must be provided right there with the instruction.

Certain 32-bit values cannot be represented as an immediate operand to a single 32-bit instruction, although you can load these values from memory in a single instruction.

MOV can load any 8-bit immediate value, giving a range of 0x0-0xFF (0-255). It can also rotate these values by any **even** number.



```
        AREA PRUEBA1, CODE, READONLY
    ENTRY
        MOV R1,#0X23
HERE    B HERE
    END
```