CMPE-250
Assembly and Embedded Programming

Introduction

- Course information
- Assembly language overview
- NXP (Freescale) KL05 architecture (ARM Cortex-M0+ core)
- Assembly language program structure

Course Information Resources

- myCourses.rit.edu
  - Resources
  - Homework
  - Lectures
  - Study Problems
  - Exams and Quizzes
  - Laboratory Exercises
- Syllabus
- Schedule
- Laboratory Report Guidelines

CMPE-250
Assembly and Embedded Programming

Introduction

- Course information
- ☞ Assembly language overview
- NXP (Freescale) KL05 architecture (ARM Cortex-M0+ core)
- Assembly language program structure

Skills for Lab Exercise One

Important questions:

- What is assembly language?
  - Language that translates directly to processor operations
- What can an assembly language program do?
  - Low-level HW control
  - Performance optimization
  - …
- What are the components of an assembly language program?
  - Assembly language program structure

Why Assembly Language?

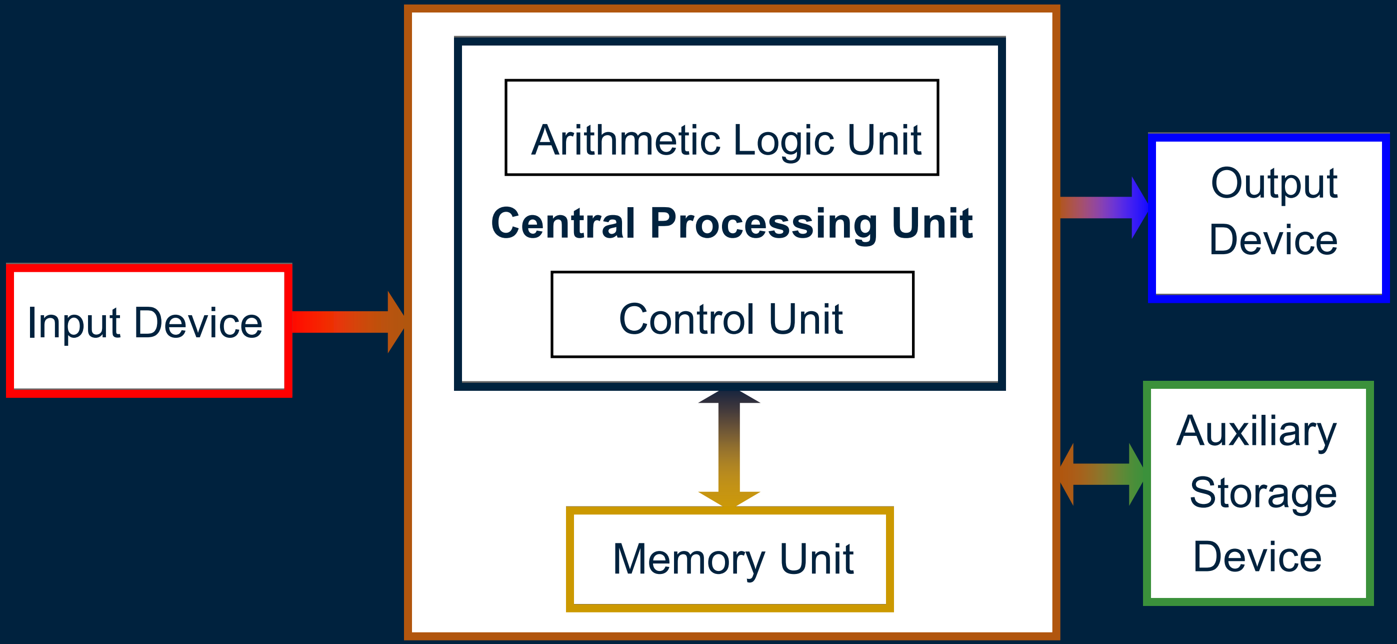
Lowest level programming

- Closest to hardware

Direct correspondence to machine instruction set

- 1 assembly language instruction = 1 machine instruction

Digital Computer Organization



Central Processing Unit (CPU)

- Arithmetic Logic Unit (ALU): Performs arithmetic and logical operations on data
- Control Unit: Controls sequence of executing program instructions and data flow in system

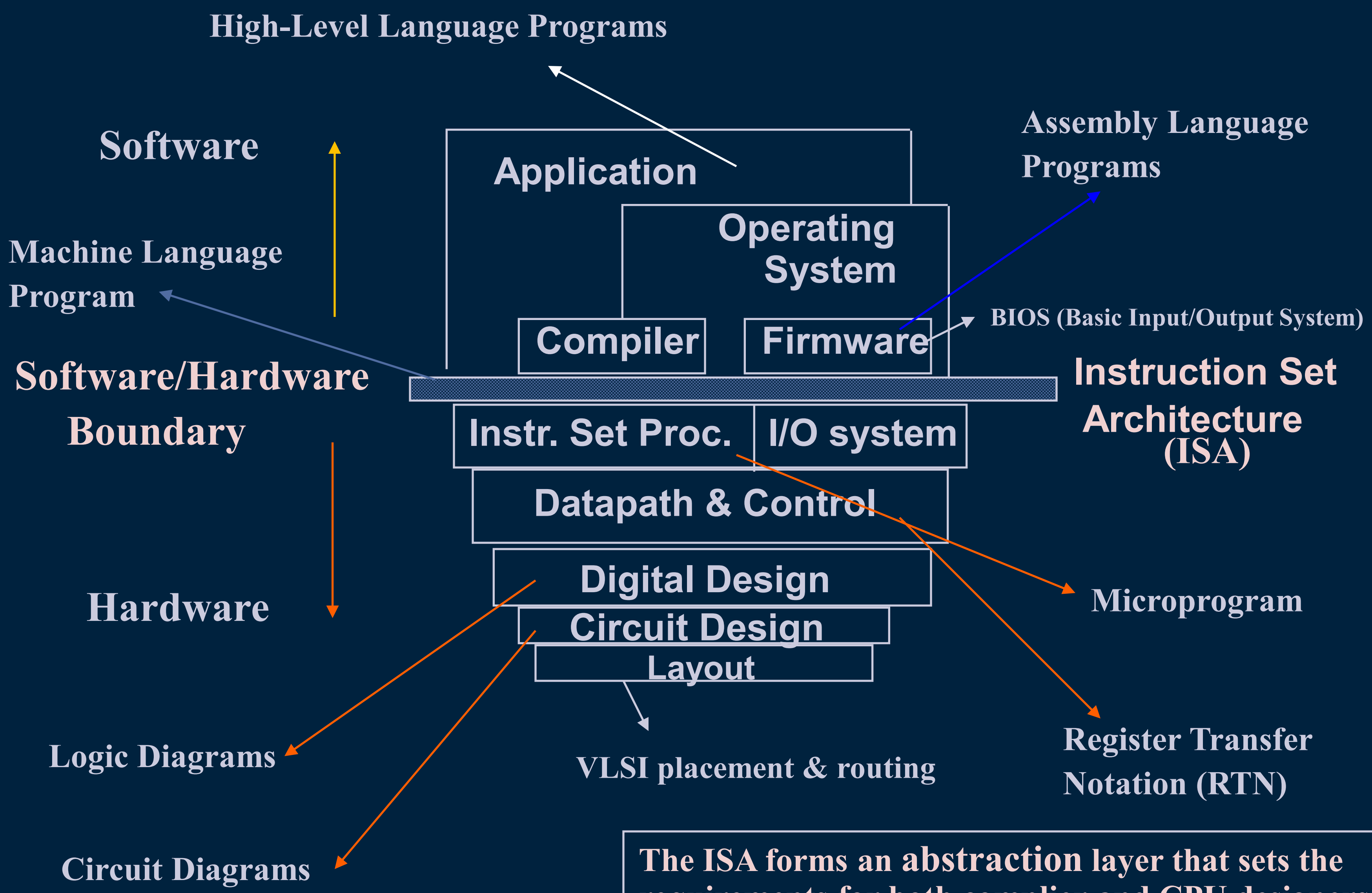
Memory Unit

Stores program instructions and data

Input and Output (I/O) Unit

Interfaces CPU with outside world: keyboard, mouse, screen, switches, LEDs, etc.

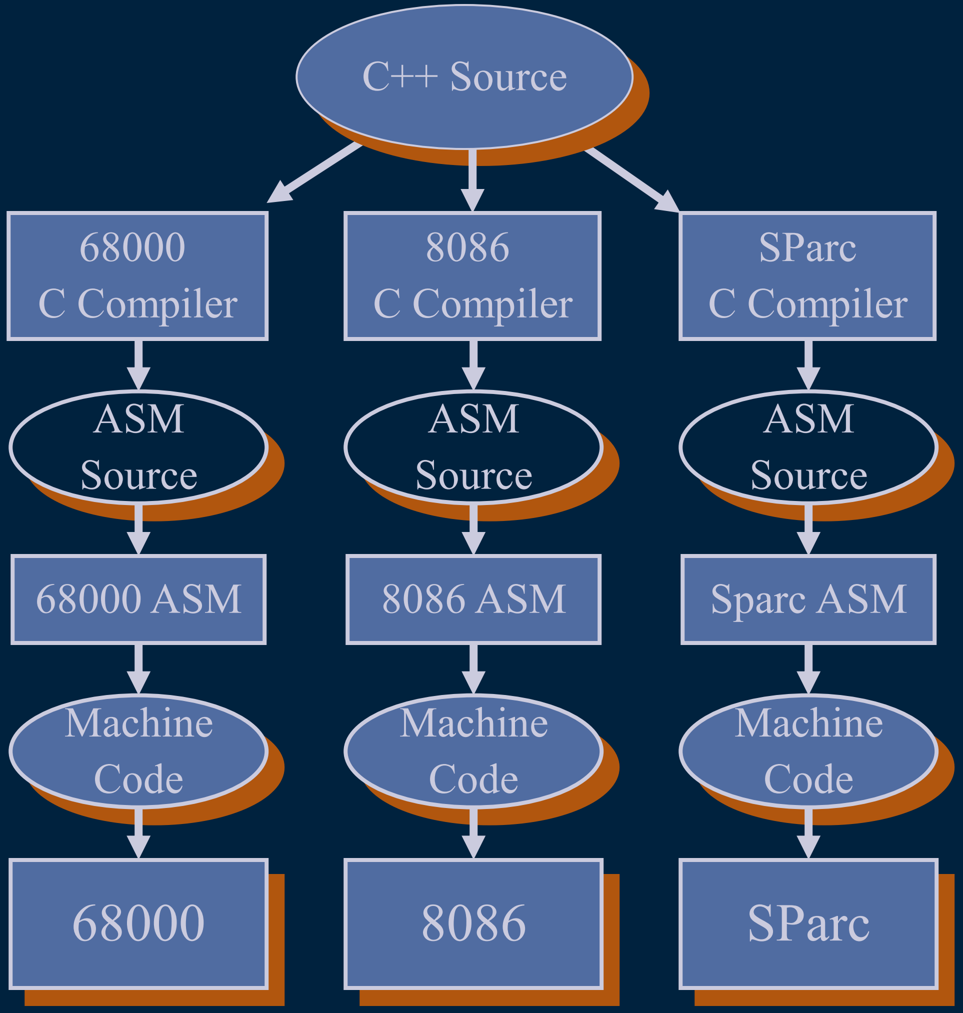
Hierarchy of Computer Architecture



The ISA forms an abstraction layer that sets the requirements for both complier and CPU designers.

Material used with permission from Dr. M. Shaaban, Dept. of Comp. Eng., RIT.

Assembly Language on Boundary Between Hardware and Software



C++ Source

68000 C Compiler

8086 C Compiler

SParc C Compiler

ASM Source

ASM Source

ASM Source

68000 ASM

8086 ASM

Sparc ASM

Machine Code

Machine Code

Machine Code

68000

8086

SParc

Material from Dr. F. Hu, EECC 250.

Assembly Language Applications

Precise hardware control

- Device drivers

System software

- Operating systems, device drivers, compilers

Precise timing (accuracy)

Memory constrained, embedded systems

Extreme performance

- Application-specific libraries
  - Graphics, scientific, etc.

Design Methodology

Flow control

Outlines

Top-down (hierarchical) design

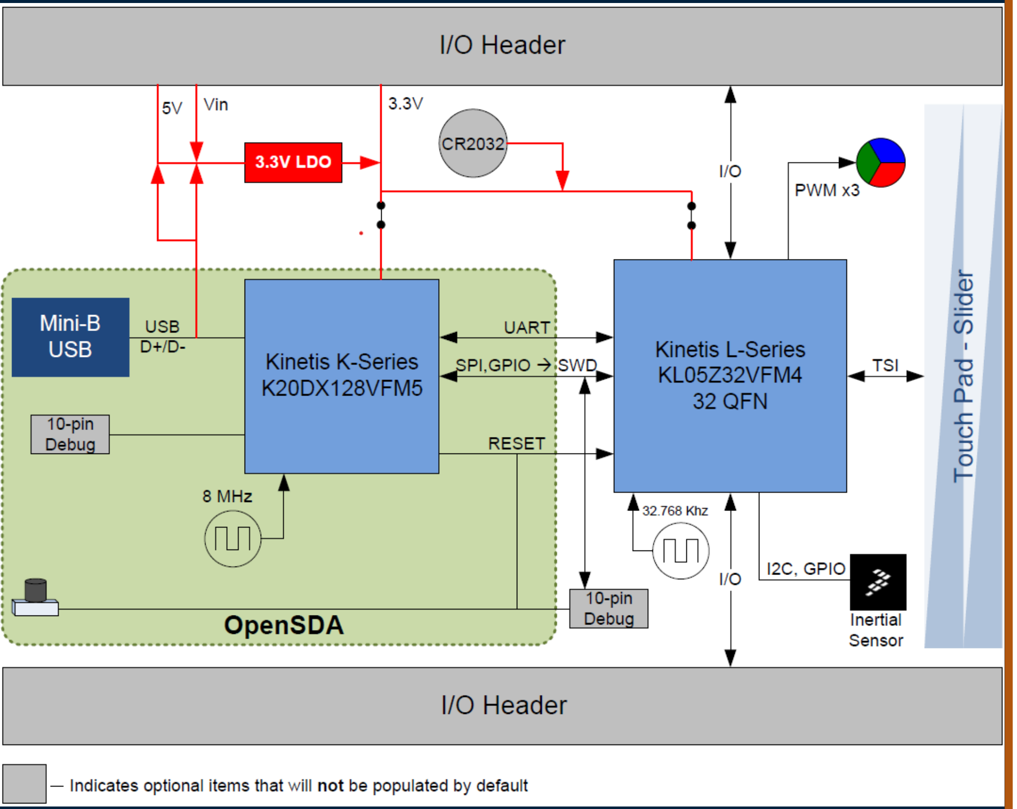
Procedural—not object-oriented

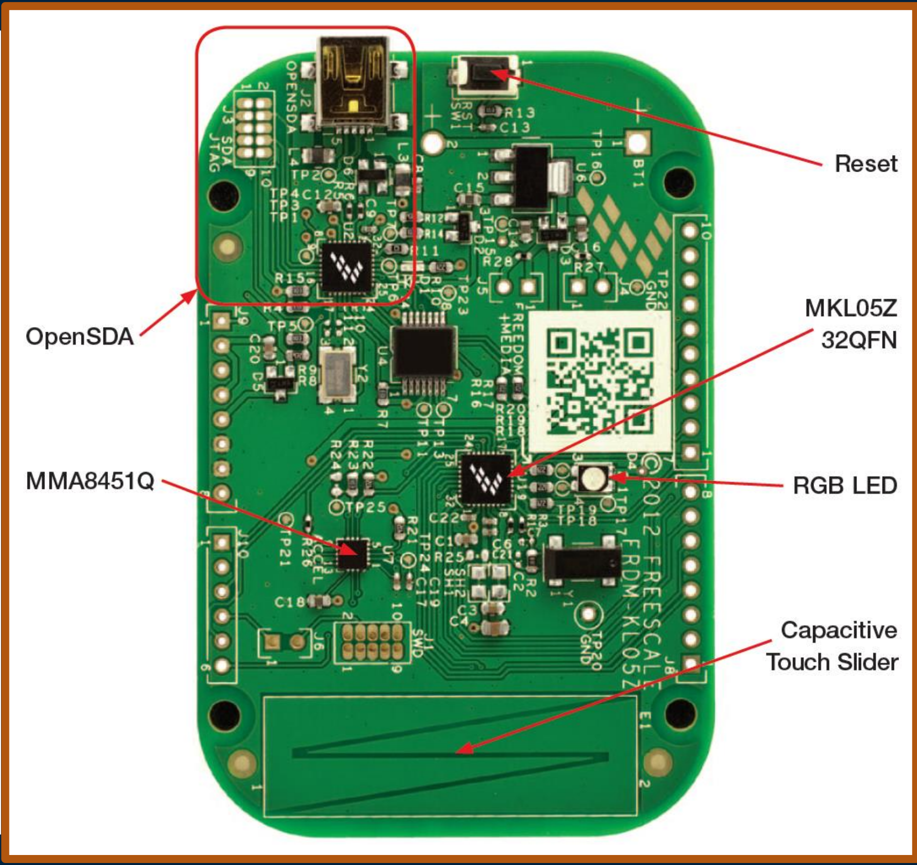
CMPE-250
Assembly and Embedded Programming

Introduction

- Course information
- Assembly language overview
- ☞ NXP (Freescale) KL05 architecture (ARM Cortex-M0+ core)
- Assembly language program structure

Freedom Development Board FRDM-KL05Z





FRDM-KL05Z User's Manual, Rev. 1.0, pp. 1, 4.

Kinetis L-Series KL05 Sub-Family

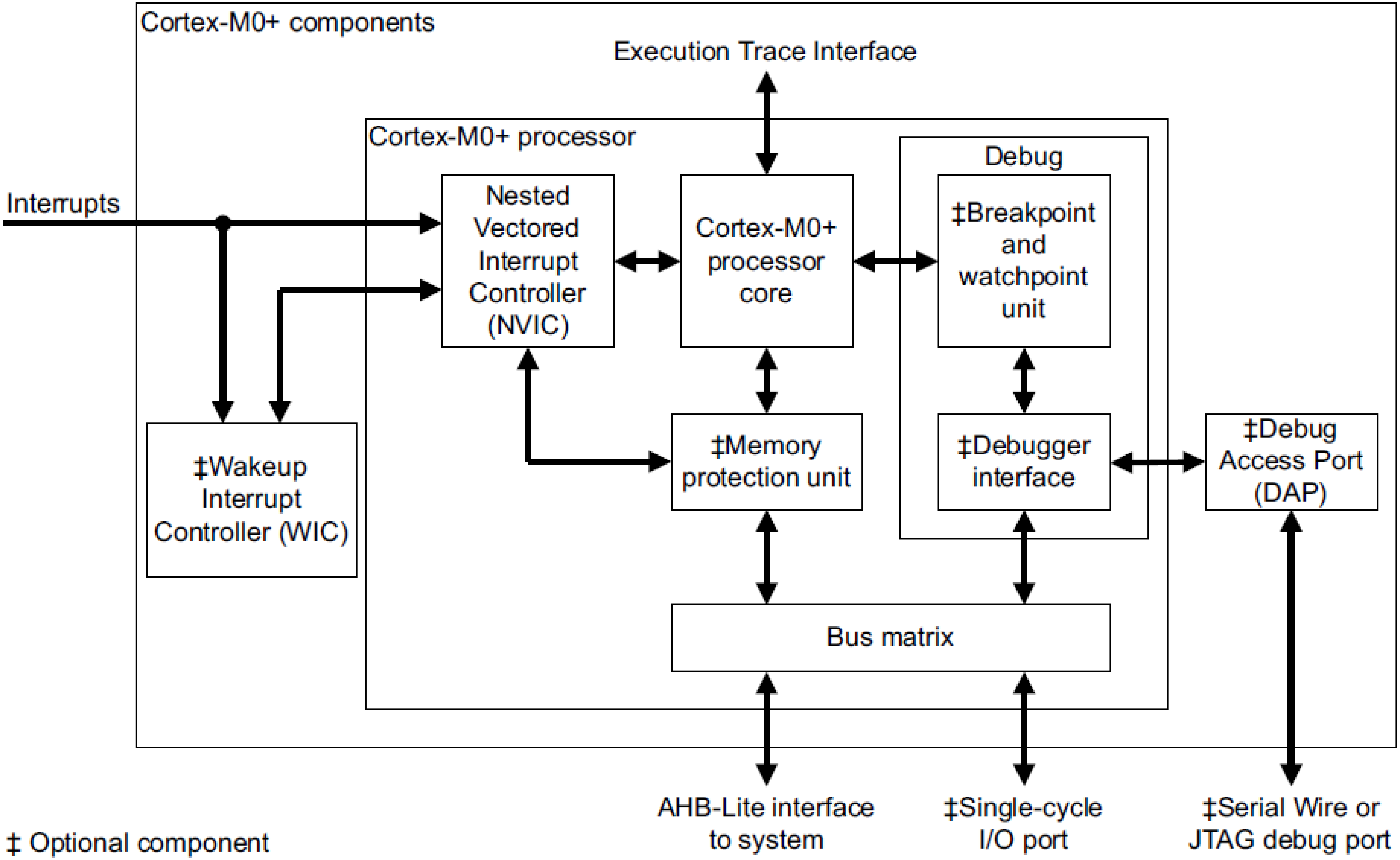
- Microcontroller MKL05Z32VFM4
  - 32 KB Flash Memory **(code, constants)**
  - 4 KB SRAM **(variables)**
- ARM Cortex-M0+ Core
  - 32-bit CPU
  - 19 CPU registers

    (Only 8 available to most instructions)
  - ARMv6-M Thumb instruction set
    - Fifty 16-bit Thumb instructions

      from ARMv7-M Thumb (1995)
    - Six 32-bit Thumb-2 instructions

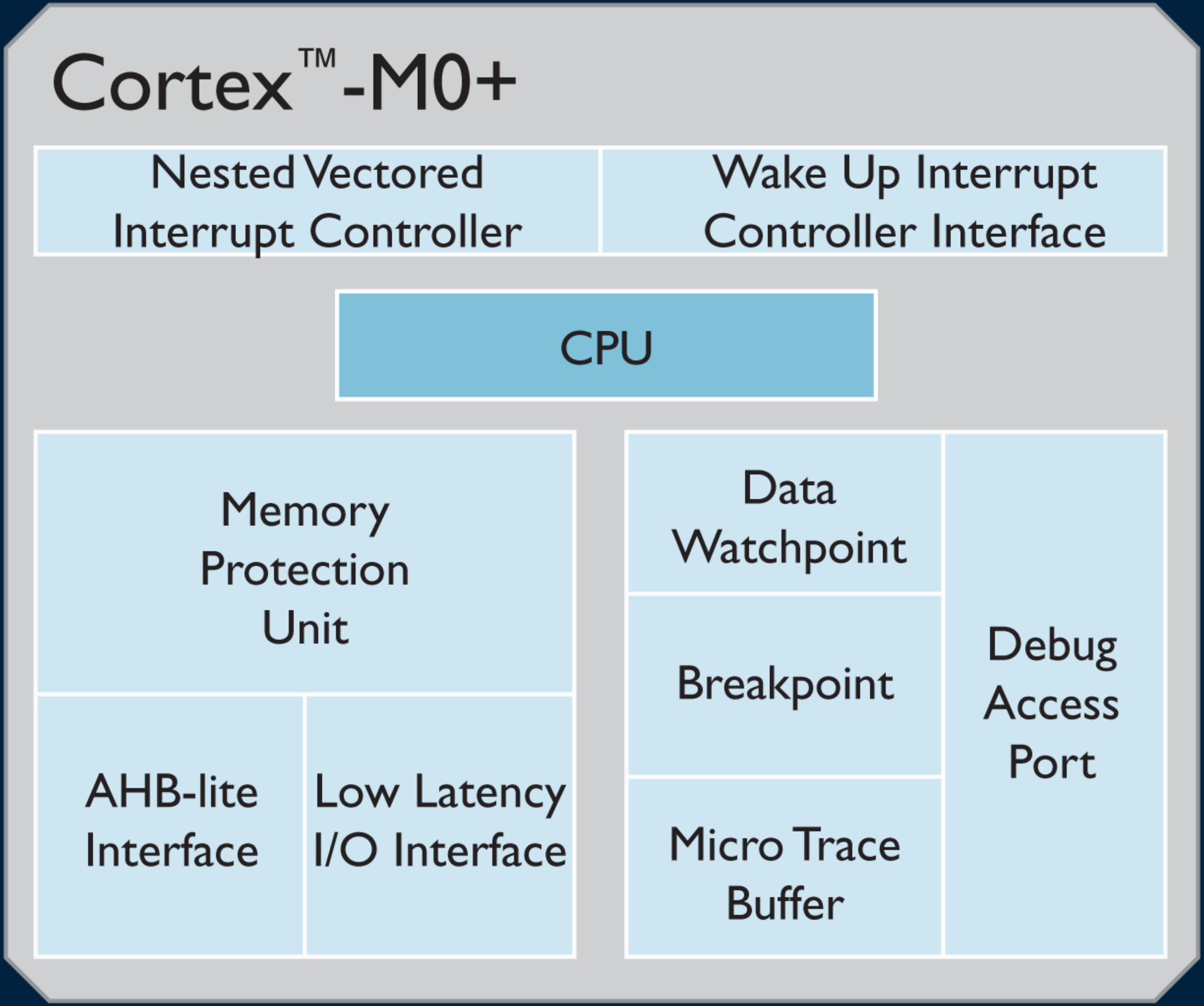
      from ARMv6-T2 (2003)

ARM Cortex-M0+ Block Diagram


Cortex-M0+ components
Execution Trace Interface
Cortex-M0+ processor
Interrupts
Nested Vectored Interrupt Controller (NVIC)
Cortex-M0+ processor core
Debug
‡Breakpoint and watchpoint unit
‡Wakeup Interrupt Controller (WIC)
‡Memory protection unit
‡Debugger interface
‡Debug Access Port (DAP)
Bus matrix
AHB-Lite interface to system
‡Single-cycle I/O port
‡Serial Wire or JTAG debug port
‡ Optional component

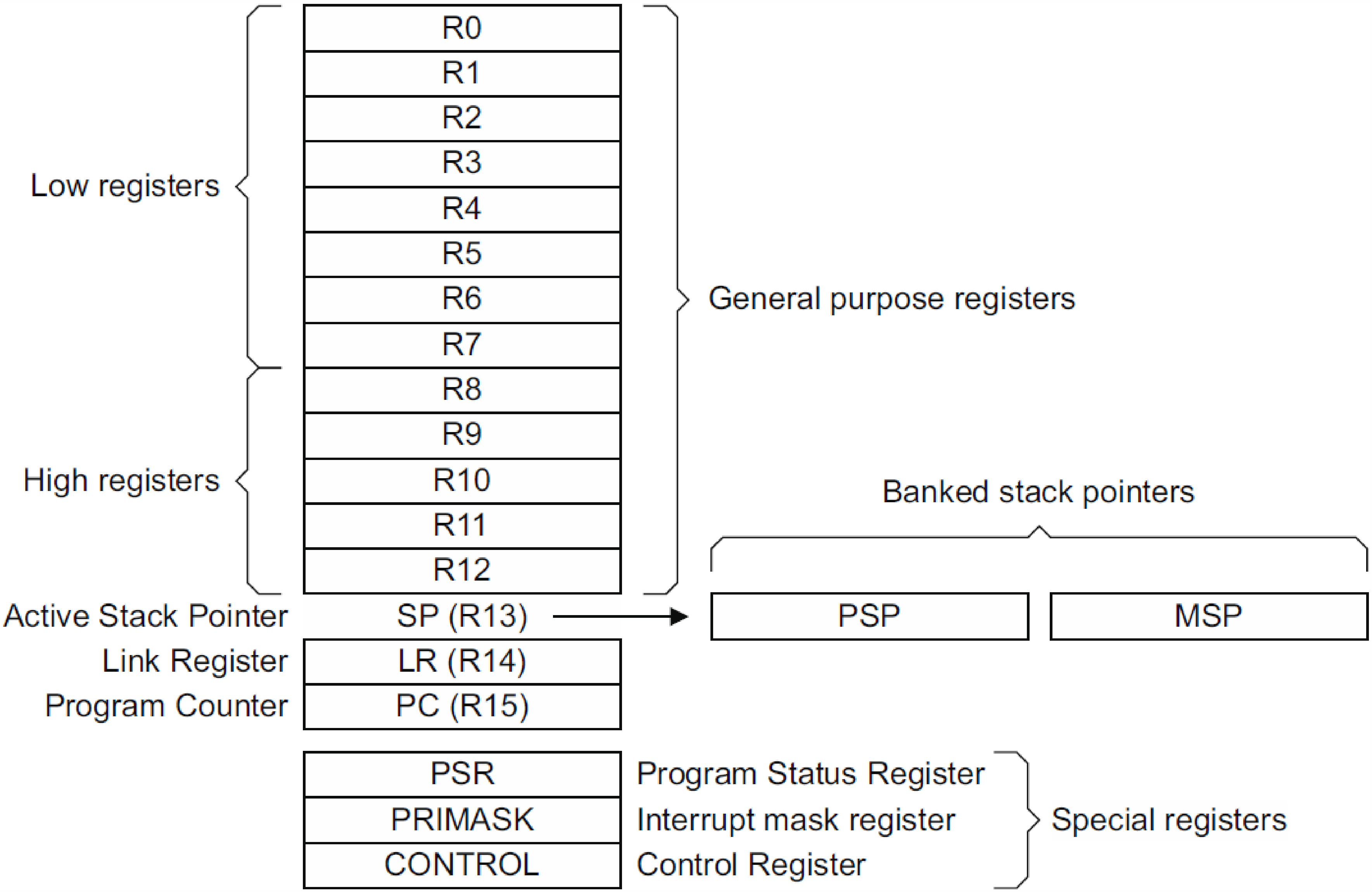
Cortex-M0+ Technical Reference Manual, p. 2-2.

Cortex-M0+ Components



http://www.arm.com/images/Cortex_M0_plus_large.png

Cortex-M0+ CPU Registers



Cortex-M0+ Devices Generic User Guide, ARM DUI 0662B (ID011713), 2012, p. 2-3.

CMPE-250
Assembly and Embedded Programming

Introduction

- Course information
- Assembly language overview
- NXP (Freescale) KL05 architecture (ARM Cortex-M0+ core)

☞ Assembly language program structure

Skills for Lab Exercise One

What is in an assembly language program?

- What is assembly language?
  - Language that translates directly to processor operations
- What can an assembly language program do?
  - NXP KL05 microcontroller
  - ARM Cortex-M0+ architecture

☞ What are the components of an assembly language program?
  - Assembly language program structure

Source: Lecture01.s (excerpt p. 1 of 4)

```
            TTL CMPE 250 Lecture 01 Example
;****************************************************************
;Example assembly language program
;Name:  R. W. Melton
;Date:  August 26, 2024
;Class:  CMPE-250
;Section:  All sections
;---------------------------------------------------------------
;Keil Simulator Template for KL05
;R. W. Melton
;August 19, 2024
;****************************************************************
;Assembler directives
            THUMB
            OPT    64  ;Turn on listing macro expansions
;****************************************************************
;EQUates
NUM_ITEM    EQU   5
```

Source: Lecture01.s (excerpt p. 2 of 4)

```
;****************************************************************
;Program
;Linker requires Reset_Handler
            AREA    MyCode,CODE,READONLY
            ENTRY
            EXPORT  Reset_Handler
Reset_Handler  PROC {}
main
;---------------------------------------------------------------
;Initialize registers R0-R12
            BL      RegInit
;>>>>> begin main program code <<<<<
            MOVS    R0,#0
            MOVS    R1,#1
            MVNS    R2,R0
            ADDS    R2,R2,#1
            MVNS    R3,R1
            ADDS    R3,R3,#1
;>>>>>   end main program code <<<<<
;Stay here
            B       .
            ENDP
;---------------------------------------------------------------
```

Source: Lecture01.s (excerpt p. 3 of 4)

```
;****************************************************************
;Vector Table Mapped to Address 0 at Reset
;Linker requires __Vectors to be exported
            AREA    RESET, DATA, READONLY
            EXPORT  __Vectors
            EXPORT  __Vectors_End
            EXPORT  __Vectors_Size
__Vectors
                                      ;ARM core vectors
            DCD    __initial_sp       ;00:end of stack
            DCD    Reset_Handler      ;reset vector
            SPACE  (VECTOR_TABLE_SIZE - (2 * VECTOR_SIZE))
__Vectors_End
__Vectors_Size  EQU     __Vectors_End - __Vectors
            ALIGN
;****************************************************************
;Constants
            AREA    MyConst,DATA,READONLY
;>>>>> begin constants here <<<<<
list        DCD     -2
            DCD     -1
            DCD     0
            DCD     1
            DCD     2
            DCD     list
            DCD     VarData
            DCD     (VarData + (NUM_ITEM * 4))
;>>>>>   end constants here <<<<<
```

Source: Lecture01.s (excerpt p. 4 of 4)

```
;****************************************************************
            AREA    |.ARM.__at_0x1FFFFC00|,DATA,READWRITE,ALIGN=3
            EXPORT  __initial_sp
;Allocate system stack
            IF      :LNOT::DEF:SSTACK_SIZE
SSTACK_SIZE EQU     0x00000100
            ENDIF
Stack_Mem   SPACE   SSTACK_SIZE
__initial_sp
;****************************************************************
;Variables
            AREA    MyData,DATA,READWRITE
;>>>>> begin variables here <<<<<
VarData     SPACE   (NUM_ITEM * 4)  ; NUM_ITEM x 4 bytes of storage
;>>>>>   end variables here <<<<<
            END
```

Listing: Lecture01.lst (excerpt p. 1 of 4)

```
ARM Macro Assembler    Page 1 CMPE 250 Lecture 01 Example


    1 00000000                 TTL              CMPE 250 Lecture 01 Example
    2 00000000         ;****************************************************************
    3 00000000         ;Example assembly language program
    4 00000000         ;Name:  R. W. Melton
    5 00000000         ;Date:  August 26, 2024
    6 00000000         ;Class:  CMPE-250
    7 00000000         ;Section:  All sections
    8 00000000         ;---------------------------------------------------------------
    9 00000000         ;Keil Simulator Template for KL05
   10 00000000         ;R. W. Melton
   11 00000000         ;August 19, 2024
   12 00000000         ;****************************************************************
   13 00000000         ;Assembler directives
   14 00000000                 THUMB
   16 00000000         ;****************************************************************
   17 00000000         ;EQUates
   18 00000000 00000005
                       NUM_ITEM
                               EQU              5
```

Listing: Lecture01.lst (excerpt p. 2 of 4)

```
128 00000000         ;****************************************************************
129 00000000         ;Program
130 00000000         ;Linker requires Reset_Handler
131 00000000                 AREA             MyCode,CODE,READONLY
132 00000000                 ENTRY
133 00000000                 EXPORT           Reset_Handler
134 00000000         Reset_Handler
                             PROC             {}
135 00000000         main
136 00000000         ;--------------------------------------------------------------
137 00000000         ;Initialize registers R0-R12
138 00000000 F7FF FFFE       BL               RegInit
139 00000004         ;>>>>> begin main program code <<<<<
140 00000004 2000            MOVS             R0,#0
141 00000006 2101            MOVS             R1,#1
142 00000008 43C2            MVNS             R2,R0
143 0000000A 1C52            ADDS             R2,R2,#1
144 0000000C 43CB            MVNS             R3,R1
145 0000000E 1C5B            ADDS             R3,R3,#1
146 00000010         ;>>>>>   end main program code <<<<<
147 00000010         ;Stay here
148 00000010 E7FE            B                .
149 00000012                 ENDP
150 00000012         ;--------------------------------------------------------------
```

Listing: Lecture01.lst (excerpt p. 3 of 4)

```
  188 00000046 00 00           ALIGN
  189 00000048         ;****************************************************************
  190 00000048         ;Vector Table Mapped to Address 0 at Reset
  191 00000048         ;Linker requires __Vectors to be exported
  192 00000048 11111111
              05240102         AREA             RESET, DATA, READONLY
  193 00000000                 EXPORT           __Vectors
  194 00000000                 EXPORT           __Vectors_End
  195 00000000                 EXPORT           __Vectors_Size
  196 00000000         __Vectors
  197 00000000         ;ARM core vectors
  198 00000000 00000000        DCD              __initial_sp ;00:end of stack
  199 00000004 00000000        DCD              Reset_Handler ;reset vector
  200 00000008 00 00 00
→ many lines of     00 00 00 omitted here
00                SPACE           (VECTOR_TABLE_SIZE - (2 * VECTOR_SIZE))
  201 000000C0         __Vectors_End
  202 000000C0 000000C0
                       __Vectors_Size
                               EQU              __Vectors_End - __Vectors
  203 000000C0                 ALIGN
  204 000000C0         ;****************************************************************
  205 000000C0         ;Constants
  206 000000C0                 AREA             MyConst,DATA,READONLY
  207 00000000         ;>>>>> begin constants here <<<<<
  208 00000000 FFFFFFFE
                       list    DCD              -2
  209 00000004 FFFFFFFF        DCD              -1
  210 00000008 00000000        DCD              0
  211 0000000C 00000001        DCD              1
  212 00000010 00000002        DCD              2
  213 00000014 00000000        DCD              list
  214 00000018 00000000        DCD              VarData
  215 0000001C 00000014        DCD              (VarData + (NUM_ITEM * 4))
```

Listing: Lecture01.lst (excerpt p. 4 of 4)

```
217 00000020        ;****************************************************************
218 00000020                AREA             |.ARM.__at_0x1FFFFC00|,DATA,READWRITE,ALIGN=3
219 00000000                EXPORT           __initial_sp
220 00000000        ;Allocate system stack
221 00000000                IF               :LNOT::DEF:SSTACK_SIZE
223                         ENDIF
224 00000000 00 00 00
→ many lines of 00 00 00 omitted here
             00     Stack_Mem
                            SPACE            SSTACK_SIZE
225 00000100        __initial_sp
226 00000100        ;****************************************************************
227 00000100        ;Variables
228 00000100                AREA             MyData,DATA,READWRITE
229 00000000        ;>>>>> begin variables here <<<<<
230 00000000 00 00 00
            00 00 00
            00 00 00
            00 00 00
            00 00 00
            00 00 00
            00 00   VarData SPACE            (NUM_ITEM * 4) ; NUM_ITEM x 4 bytes of storage
231 00000014        ;>>>>>   end variables here <<<<<
232 00000014                END
```

Map: Lecture01.map (page 1 of 3)

```
Component: ARM Compiler 5.06 update 7 (build 960) Tool: armlink [4d3601]
==============================================================================
Section Cross References
    lecture01.o(RESET) refers to lecture01.o(.ARM.__at_0x1FFFFC00) for __initial_sp
    lecture01.o(RESET) refers to lecture01.o(MyCode) for Reset_Handler
    lecture01.o(MyConst) refers to lecture01.o(MyData) for VarData
==============================================================================
Removing Unused input sections from the image.
    Removing lecture01.o(MyConst), (32 bytes).
    Removing lecture01.o(MyData), (20 bytes).
2 unused section(s) (total 52 bytes) removed from the image.
==============================================================================
Image Symbol Table
    Local Symbols
    Symbol Name                              Value     Ov Type        Size  Object(Section)
    RESET                                    0x00000000   Section      192  lecture01.o(RESET)
    Lecture01.s                              0x00000000   Number         0  lecture01.o ABSOLUTE
    MyCode                                   0x000000c0   Section       80  lecture01.o(MyCode)
    RegInit                                  0x000000d3   Thumb Code    52  lecture01.o(MyCode)
    .ARM.__at_0x1FFFFC00                     0x1ffffc00   Section      256  main.o(.ARM.__at_0x1FFFFC00)
Global Symbols
    Symbol Name                              Value     Ov Type        Size  Object(Section)
    __Vectors                                0x00000000   Data           0  lecture01.o(RESET)
    Reset_Handler                            0x000000c1   Thumb Code    18  lecture01.o(MyCode)
    __Vectors_End                            0x000000c0   Data           0  lecture01.o(RESET)
    __Vectors_Size                           0x000000c0   Number         0  lecture01.o ABSOLUTE
    __initial_sp                             0x1ffffd00   Data           0  main.o(.ARM.__at_0x1FFFFC00)
```

Map: Lecture01.map (page 2 of 3)

```
==============================================================================
Memory Map of the image

  Image Entry point : 0x000000c1

  Load Region LR_1 (Base: 0x00000000, Size: 0x00000210, Max: 0xffffffff, ABSOLUTE)

    Execution Region ER_RO
      (Exec base: 0x00000000, Load base: 0x00000000, Size: 0x00000110, Max: 0xffffffff, ABSOLUTE)

    Exec Addr    Load Addr    Size         Type   Attr       Idx    E Section Name        Object

    0x00000000   0x00000000   0x000000c0   Data   RO              2    RESET               lecture01.o
    0x000000c0   0x000000c0   0x00000050   Code   RO              1  * MyCode              lecture01.o


    Execution Region ER_RW
      (Exec base: 0x1ffffc00, Load base: 0x00000110, Size: 0x00000100, Max: 0xffffffff, ABSOLUTE)

    Exec Addr    Load Addr    Size         Type   Attr       Idx    E Section Name        Object

    0x1ffffc00   0x00000108   0x00000100   Data   RW              4    .ARM.__at_0x1FFFFC00  main.o


    Execution Region ER_ZI
      (Exec base: 0x1ffffd00, Load base: 0x00000210, Size: 0x00000000, Max: 0xffffffff, ABSOLUTE)

    **** No section assigned to this execution region ****
```

Map: Lecture01.map (page 3 of 3)

```
==============================================================================
Image component sizes
      Code (inc. data)   RO Data    RW Data    ZI Data      Debug   Object Name
        80         10        192        256          0        392   lecture01.o
    ----------------------------------------------------------------------
        80         10        192        256          0        392   Object Totals
         0          0          0          0          0          0   (incl. Generated)
         0          0          0          0          0          0   (incl. Padding)
    ----------------------------------------------------------------------
         0          0          0          0          0          0   Library Totals
         0          0          0          0          0          0   (incl. Padding)
    ----------------------------------------------------------------------
==============================================================================
      Code (inc. data)   RO Data    RW Data    ZI Data      Debug
        80         10        192        256          0        392   Grand Totals
        80         10        192        256          0        392   ELF Image Totals
        80         10        192        256          0          0   ROM Totals
==============================================================================
    Total RO  Size (Code + RO Data)                      272 (   0.27kB)
    Total RW  Size (RW Data + ZI Data)                   256 (   0.25kB)
    Total ROM Size (Code + RO Data + RW Data)            528 (   0.52kB)
==============================================================================
```