Chapter 3: Development tools and software support

Stellaris® Cortex™-M3 - Microcontroller Family

Texas Instruments

Texas Instruments - University Program

Heilbronn University, Campus Künzelsau

Prof. Dr.-Ing. Ralf Gessler

Rev. 1.0

Content

- Chapter 3: Development tools and software support
  - 3.1 Evaluation boards
  - 3.2 Software development tools
  - 3.3 C language: Introduction
  - 3.4 StellarisWare®

- **Topics:** development process, design flow, design elements, C versus assembler, Embedded C, C instructions and operations

Learning Objectives

- The chapter is a practically oriented introduction to the C high-level language.
- The main focus is the development of Embedded Systems based high-level languages.
- Structure and questions:
  - What is a Development Process?
  - How does a typical Design flow look?
  - What are the Design Elements?
  - What are the instructions and operators for writing a C program?
  - What are the requirements for modelling Embedded Systems

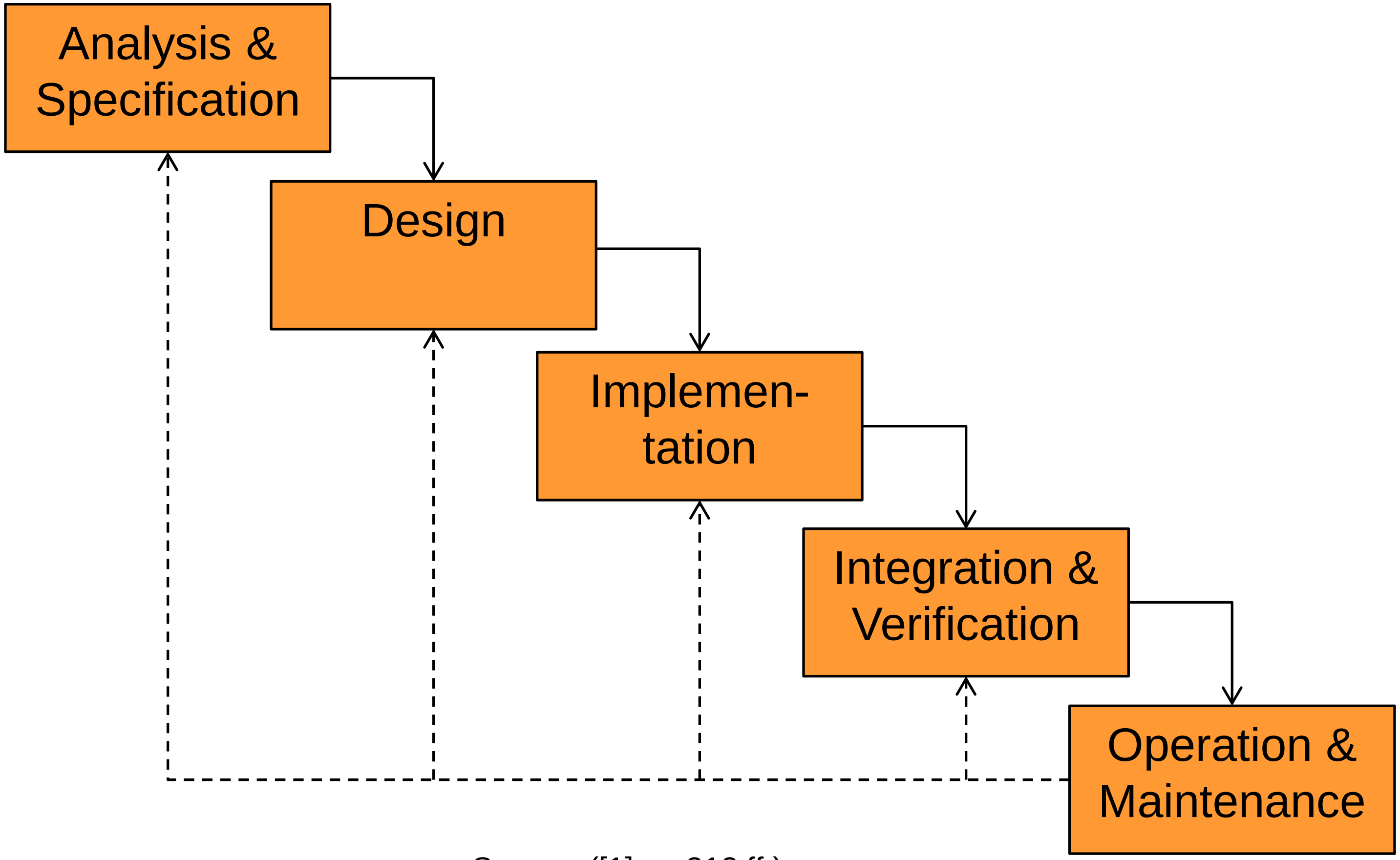
Introduction

- C was introduced in 1972 by Dennis Richie ([2], p. 62 ff.)
- Designed as programming language for UNIX
- Two important dialects:
  - K&R:
    - Kernighan & Richie style
    - Language launch
    - Obsolete
  - ANSI C:
    - Current and most significant dialect
- Characteristics
  - High level language
    - Abstract processor view
  - High availability
  - Close to hardware
  - Control and data structures

Development Process I

HOCHSCHULE HEILBRONN
HEILBRONN UNIVERSITY
ENGINEERING BUSINESS INFORMATICS
Campus Künzelsau
Reinhold-Würth-Hochschule

- Water fall model
  - A simple model



Source: ([1], p. 312 ff.)

Development Process II

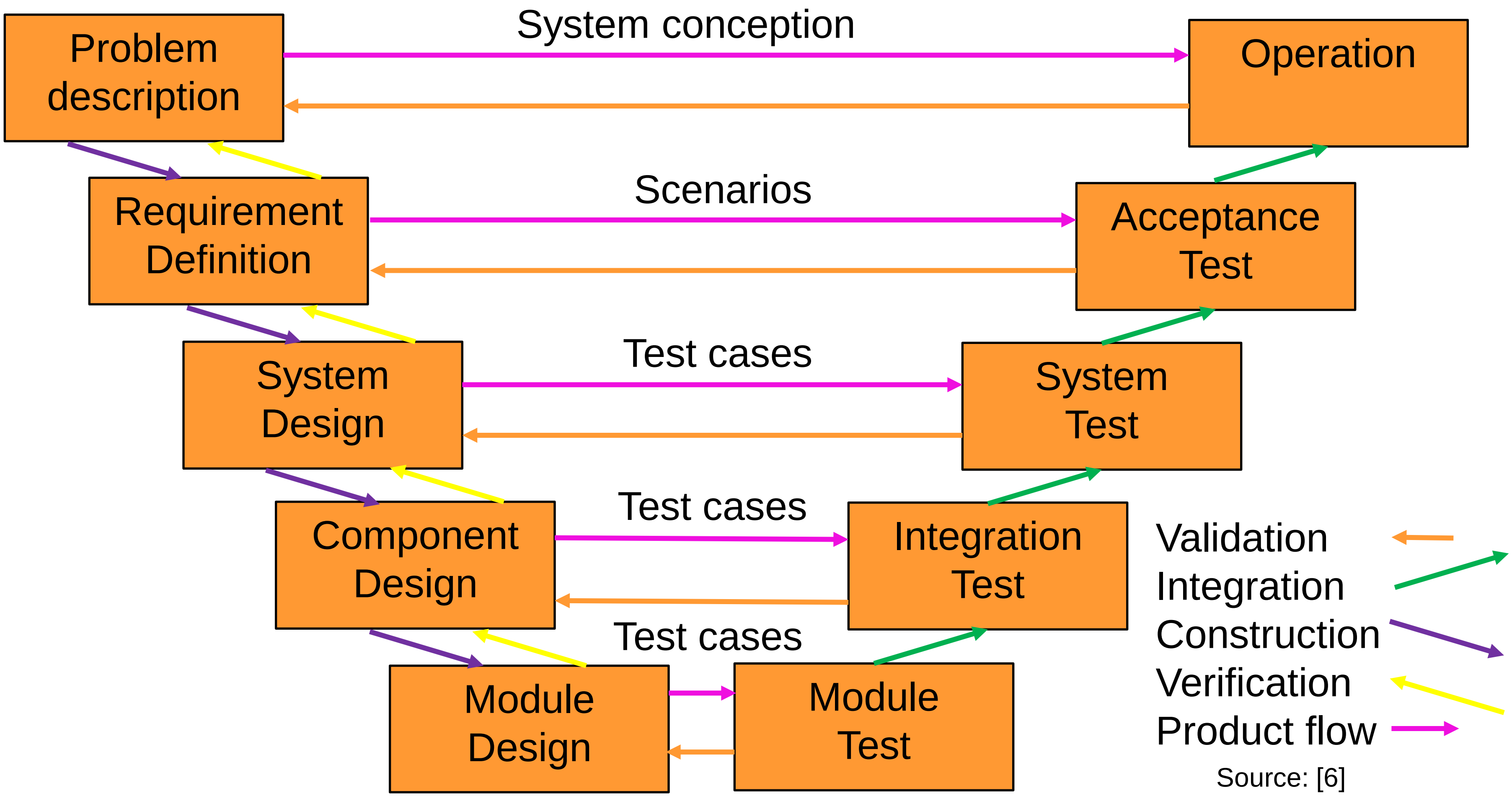
HOCHSCHULE HEILBRONN
HEILBRONN UNIVERSITY
ENGINEERING BUSINESS INFORMATICS
Campus Künzelsau
Reinhold-Würth-Hochschule

- Analysis & Specification
  - Problem analysis with constraints
  - Requirements Specification
- Design
  - System architecture
  - Hardware/Software Partitioning
- Implementation
  - Module realization
  - Subsystem (module) tests
- Integration & Verification
  - Module Composition
  - System tests
- Operation & Maintenance
  - System launch
  - Functional extension

Main focus

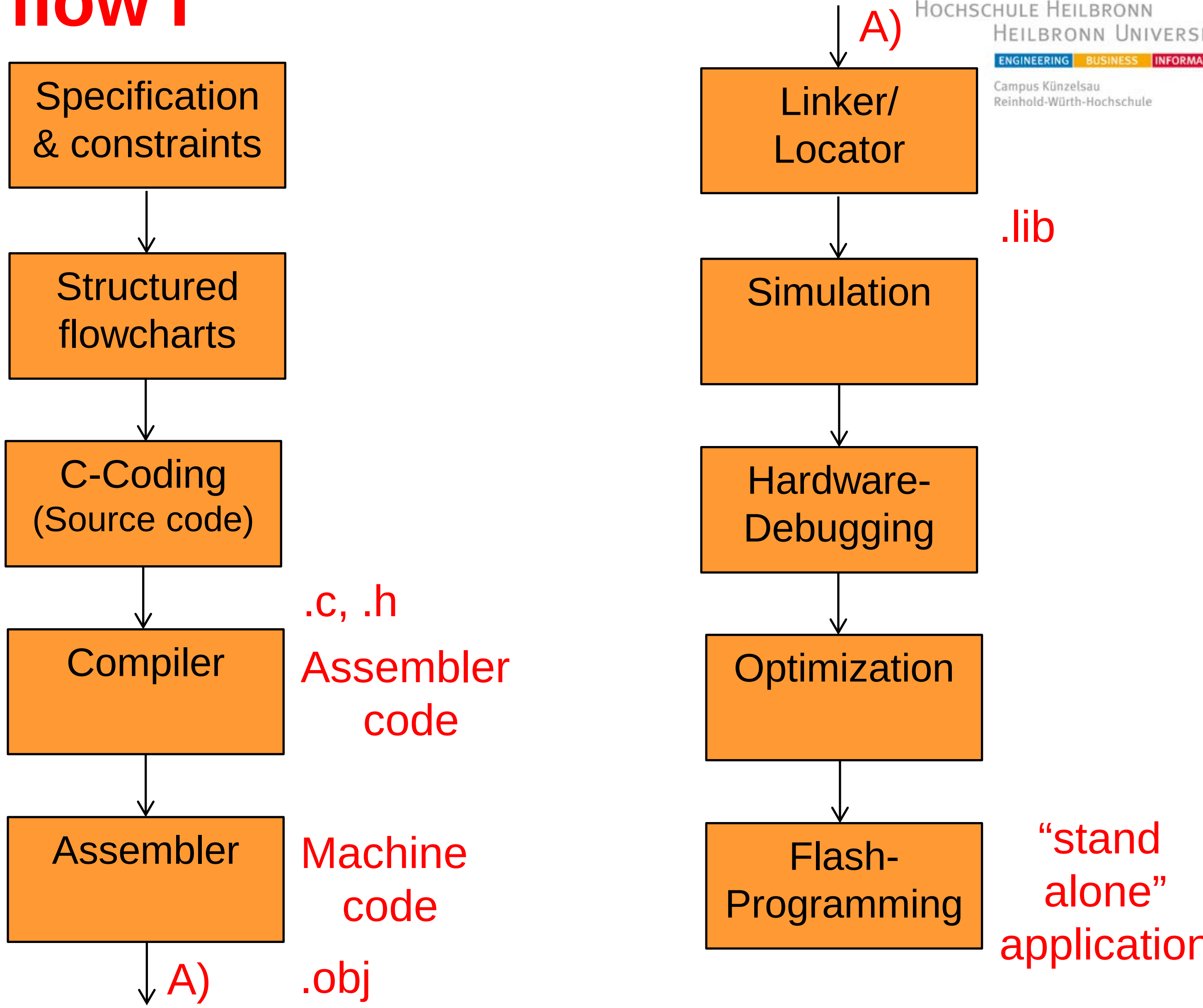
Development Process III: V-Model

- V-Model: Approach model - in German “Vorgehens-Modell”



Source: [6]

Design flow I



HOCHSCHULE HEILBRONN
HEILBRONN UNIVERSITY
ENGINEERING BUSINESS INFORMATICS
Campus Künzelsau
Reinhold-Würth-Hochschule

Design flow II

- Compiler ([1], p. 335 ff.)
  - Translation of high level instructions to Assembler or Machine code
  - Machine code (Native code): sequence of CPU instructions (binary data)
- Assembler
  - Mnemonics: 1:1 representation of machine code
- Linker/Locator
  - Module binding and attaching physical address
- Optimization
  - Optimizing tool for speed or area
- Simulator
  - Source code testing on host PC
- Debugging
  - Source code testing on target
  - Levels: source level, machine code

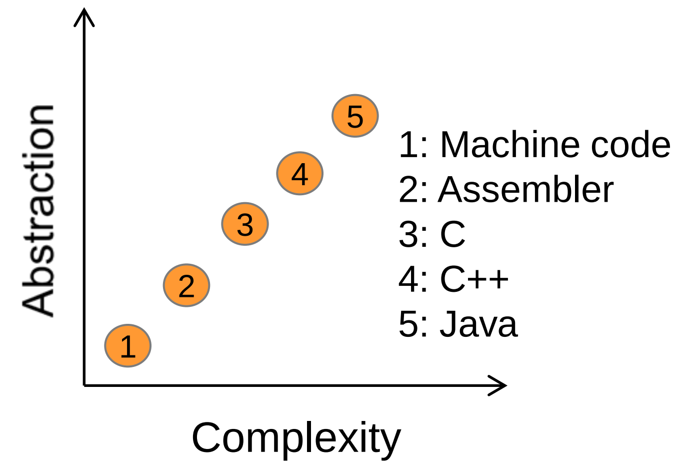
Design flow III

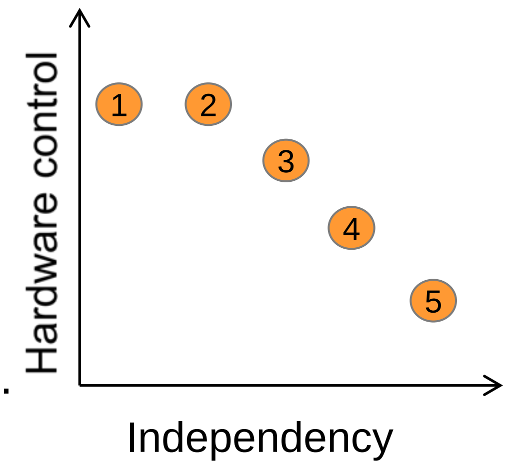
- IDE – Integrated Development Environment
  - Complete development frame including translation and test tools
- Files:
  - Source code: .c:
    - Function implementation
    - Definition global variables
  - Header files: .h
    - Declarations: functions, data types
  - .obj: object file
  - .lib: library file

C versus Assembler

HHN
Hochschule Heilbronn
Heilbronn University
ENGINEERING BUSINESS INFORMATICS
Campus Künzelsau
Reinhold-Würth-Hochschule

- C ([1], p. 333 ff. ; [4])
  - Pros
    - Higher abstraction
    - Portable
    - Better readable
    - Early syntax check
  - Cons
    - Less performance than Assembler
- Assembler
  - Pros
    - Closer hardware control
    - Commonly faster and less resources
  - Cons
    - More prone to errors. Extensive testing required.
- ⇨ Focus: C language + StellarisWare®





Source: ([4], p. 156 ff.)

Embedded C

- C language is general purpose ([5], p. 29 ff.)
  - Developing programs for data processing or embedded systems
  - Embedded software often must have immediate hardware access to internal peripherals
  - Remember: Embedded Systems
    - Link: Chapter 1.1 “Introduction and Microcontrollers”

- For the different requirements of the applications
  - An add on or a reduction of the general C language is necessary
  - This is called “Embedded C”

- This has consequences to the main advantage of a high level language: the Portability

- Microcontroller-specific categories:
  - Memory: program and data memory
  - Peripherals: Inputs/Outputs, timers etc.
  - Interrupts: sources, priority

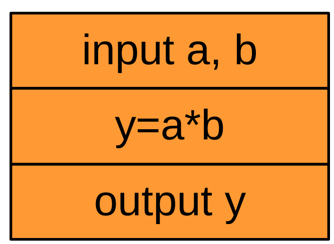
Design elements

- Pseudo code
- Flowchart
- Structured flowcharts (Nassi Shneiderman diagrams)

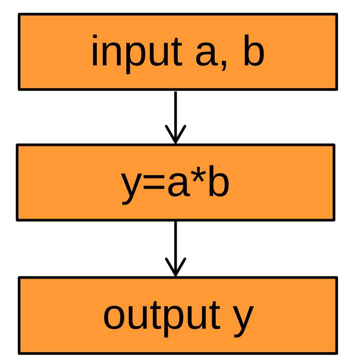
Pseudo code

y=a*b

Structured flowchart



Flowchart



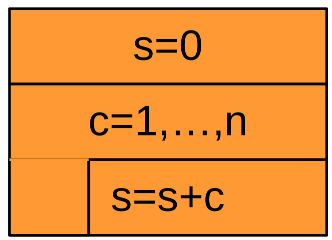
Source: [3]

Design elements: loops

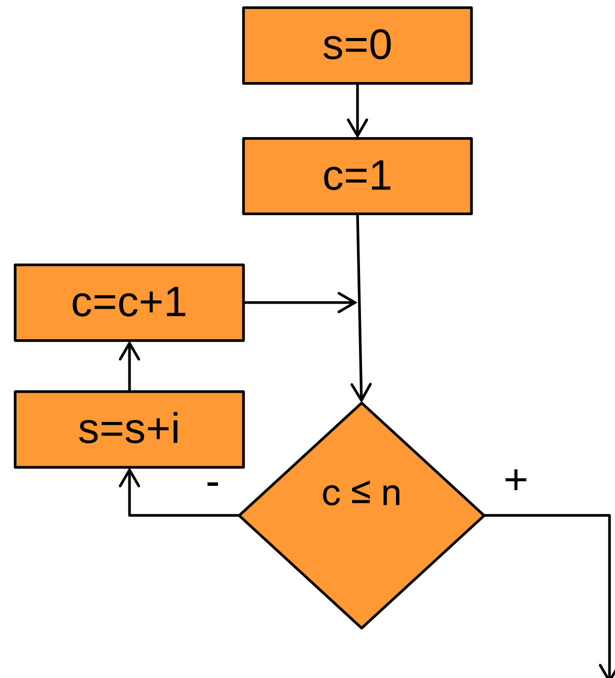
Pseudo code

```
s=0
for c=1,…,n
  s=s+c
end
```

Structured flowchart



Flowchart



+ : true
- : false

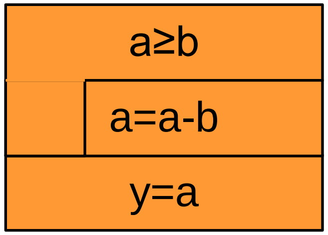
Source: [3]

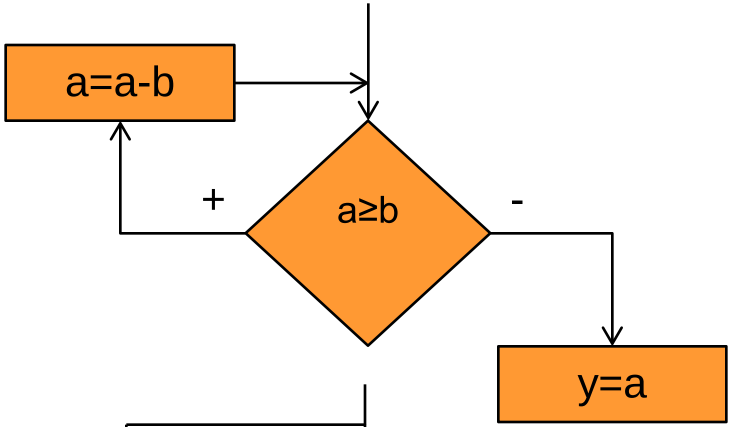
Design elements: loops

Pseudo code | Structured flowchart | Flowchart

Example: y mod b

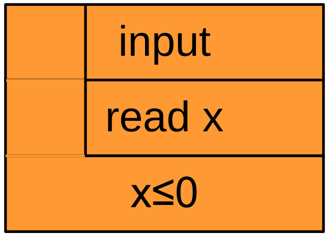
while a≥b
  a=a-b
end
y=a

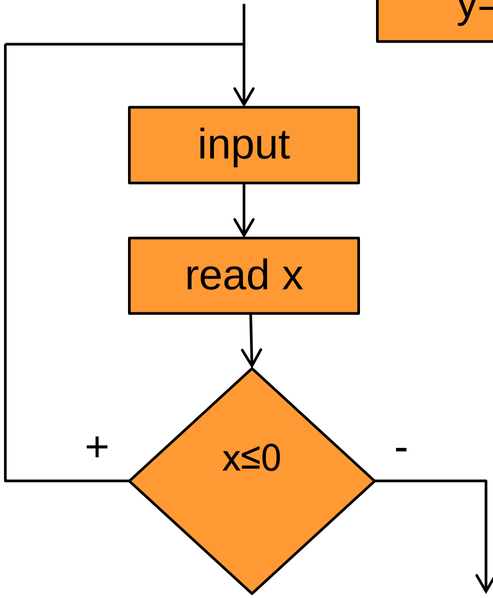




Example: input x>0

do
  input
  read x
while x≤0





Alternative: repeat…until

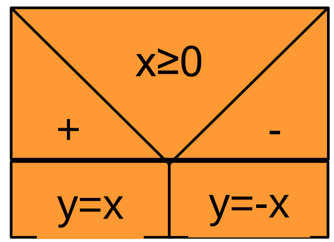
Source: [3]

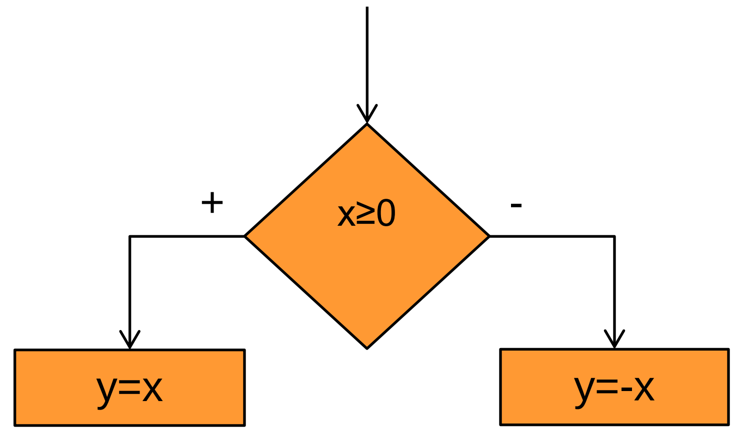
Design elements: conditions

Pseudo code | Structured flowchart | Flowchart

Example: y= x, if x≥0
-x, otherwise

if x≥0
  y=x
else
  y=-x
end

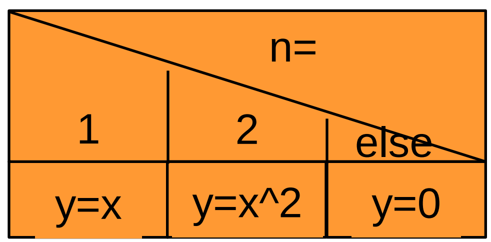


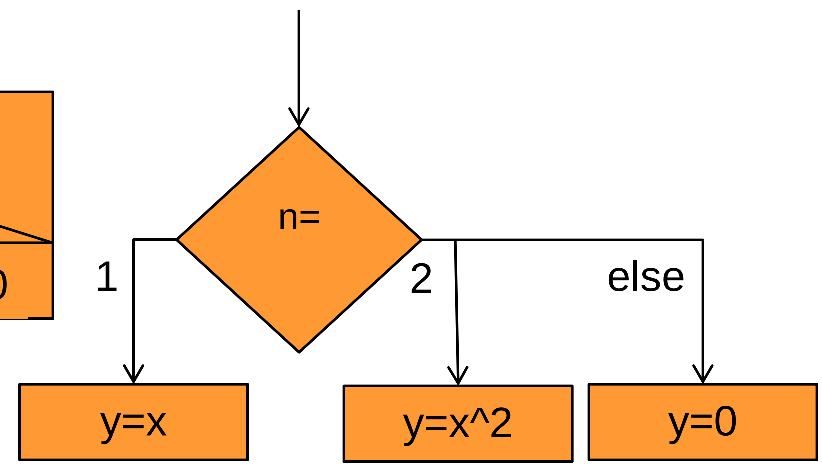


Example: y= x, if n=1
x^2, if n=2,
0, otherwise

switch n=
  case 1: y=x
  case 2: y^2
  else   y=0

Alternative:
if…else





Source: [3]

Pre-processor

HOCHSCHULE HEILBRONN
HEILBRONN UNIVERSITY
ENGINEERING BUSINESS INFORMATICS
Campus Künzelsau
Reinhold-Würth-Hochschule

- C uses an upstream pre-processor
- Characteristics
  - C independent
  - Starting with #
  - Instruction ends without semicolon
  - Erases comments
- simple macros
  - #include <name.h> (header file)
    - Default path for name.h
  - #define
    - Example: #define GPIO_PORTG_DATA_BITS_R ((volatile unsigned long *) 0x40026000)
  - #pragma
    - Pragma directives control the Compiler for implementation specific operations.

Data types

- Integer
  - Char: ASCII, 8 bits
  - Short: 8 bits
  - Int: 16 bits
  - Long: 32 bits
  - Note: compiler/architecture specific identifiers
- Floating point types
  - Float: 32 bits
  - Double: 64 bits
  - Long double: 80 bits
- Arrays:
  - Example: int adc_val [5] = {1,2,3,4,5};

Operations

Operator	Description	Example
+, -, *, /, %	addition, subtraction multiplication, division, Modulo	int y=a+b;
+, -	Sign	int y=-a;
++, --	increment, decrement	int y=a++;
&, |	bit and, or	int y= a & 0x01;
~	bit negation	int y=~a;
^	bit exclusive or	int y=a^0x01;
<<, >>	bit shift left, right	int y=a<<4;
&&, ||	logic or, and	int y= a || 0x01;
!	logic not	int y=(!(a==b))
==, !=	equal, not equal	if (y==0)
<, >, <=, >=	comparison	if (y<0)
=, +=, &=, …		int y=5;
*, &	pointer, address	int y=*a;

Instructions

Instructions	Description	Example

if (cond)
instr
else
instr
	conditional statements	if (a==0)
		y=0;
		else
		y=1;

switch (n) {
case cond1: instr1;
case cond2: instr2;
default: instr3; }
	switch statements
	multiway branches
		switch (n) {
		case 1: y=x;
		case 2: y=x^2;
		default: y=0; }

for (init; cond; incr) instr.	number of iteration is known	for (i=0;i<5;i++) {y=y+1;}
while (cond) instr	pre-test loop	while (a>0) y++;

do
intr
while (cond)
	post-test loop	do
		y++;
		while (a<0)

{instr1; instr2; instr3; …}	block	{a++; b--;}

Example: Disassembly

HHN
Hochschule Heilbronn
Heilbronn University
ENGINEERING BUSINESS INFORMATICS
Campus Künzelsau
Reinhold-Würth-Hochschule

- Basic instructions
  - For …
  - If … else …
- Link: see Chapter 2.3 “Cortex™-M3: Programmer model (ISA)”
- Link: see Lab “lab33a.zip”

Disassembly (main)

```
25                        for (i=0;i<100;i++)
                          main, main:
0x00000298:   4914        LDR         R1, $C$CON1
0x0000029A:   2000        MOV         R0, #0x0
0x0000029C:   6008        STR         R0, [R1]
0x0000029E:   4813        LDR         R0, $C$CON1
0x000002A0:   6800        LDR         R0, [R0]
0x000002A2:   2864        CMP         R0, #0x64
0x000002A4:   D215        BCS         $C$DW$L$main$2$E
27                                k=i*i;
                          $C$DW$L$main$2$B, $C$L1:
0x000002A6:   4811        LDR         R0, $C$CON1
0x000002A8:   6801        LDR         R1, [R0]
0x000002AA:   4810        LDR         R0, $C$CON1
0x000002AC:   6800        LDR         R0, [R0]
0x000002AE:   4348        MUL         R0, R1
0x000002B0:   490F        LDR         R1, $C$CON2
0x000002B2:   6008        STR         R0, [R1]
28                                y[i]=k;
0x000002B4:   480D        LDR         R0, $C$CON1
0x000002B6:   6801        LDR         R1, [R0]
0x000002B8:   480D        LDR         R0, $C$CON2
0x000002BA:   4A0E        LDR         R2, $C$CON3
0x000002BC:   6800        LDR         R0, [R0]
0x000002BE:   F8420021    STR.W       R0, [R2, R1, LSL #2]
25                        for (i=0;i<100;i++)
0x000002C2:   490A        LDR         R1, $C$CON1
0x000002C4:   6808        LDR         R0, [R1]
0x000002C6:   1C40        ADD         R0, R0, #0x1
0x000002C8:   6008        STR         R0, [R1]
0x000002CA:   4808        LDR         R0, $C$CON1
0x000002CC:   6800        LDR         R0, [R0]
0x000002CE:   2864        CMP         R0, #0x64
0x000002D0:   D3E9        BCC         $C$L1
31                        if   (i==99)
                          $C$L2, $C$DW$L$main$2$E:
0x000002D2:   4806        LDR         R0, $C$CON1
0x000002D4:   6800        LDR         R0, [R0]
0x000002D6:   2863        CMP         R0, #0x63
0x000002D8:   D103        BNE         $C$L3
32                                k=100;
0x000002DA:   4905        LDR         R1, $C$CON2
0x000002DC:   2064        MOV         R0, #0x64
0x000002DE:   6008        STR         R0, [R1]
0x000002E0:   E002        B           $C$L4
34                                k=0;
                          $C$L3:
0x000002E2:   4903        LDR         R1, $C$CON2
0x000002E4:   2000        MOV         R0, #0x0
0x000002E6:   6008        STR         R0, [R1]
36                        }
                          $C$L4:
```

Functions

- function
  - "Stand alone" block
  - Called by other blocks
  - Getting parameters (void: no parameter (obsolete))
  - Deliver max. one result (void: no result)

- master function: main()
  - Output
    - Void main(): no return
    - Int main(): integer value back (via return)
  - Input
    - Main (int argc): argc – number input parameter plus one
    - Main (int argc, char *argv[]): argv – pointer array of parameter

C Program Frame

HOCHSCHULE HEILBRONN
HEILBRONN UNIVERSITY
ENGINEERING BUSINESS INFORMATICS
Campus Künzelsau
Reinhold-Würth-Hochschule

```
/* Simple C Frame
	Project:
	Description
	Date:
	Revision:
	…
*/
#include <xxx.h>
int main (int argc, char* argv[] ) {
		instructions;
		{} // blocks

		return 0;
		}
```

Data Storage

- Variables characteristics
  - Local variables
    - Automatic variable generation by block entrance and deletion after leaving
    - Local: stack
      - Example:
        void func() {
        int localvar = 10;}
    - Local: register
      - Example:
        void func() {
        register int localvar = 10;}
  - Static and global variables
    - Static: defined inside a block, the value is conserved
    - Global: definition and initialisation outside functions
      - Extern: indicates that the global variable is still defined in a other file
  - Const: constant value, cannot be changed
  - Volatile: prevents optimisation

Data Storage

- Pointer
  - Variables containing address of other variables
  - Pointer type *
  - Address operator &

- Example:
  - Int *int_Ptr;
    int int_val = 10;
    int int_Ptr = &int_val;

Questions and Exercises

1. Explain the terms Compiler, Assembler, Linker.
2. Design a water fall model.
3. Design a V model.
4. Draw a structured flowchart for an arithmetic mean with n=5.
5. Explain the design flow for the arithmetic mean.

Summary and Outlook

- Summary
  - From Development Process to Design flow
  - From Design flow to Design elements
  - From Design elements to C frame with instructions
- Outlook/How to go on?
  - Chapter 3.2 “Software development tools”
    illustrates the practical C application.
  - Chapter 3.4 “StellarisWare®”
    presents the high-level approach with Peripheral Driver Libraries

References

- [1] Beierlein, Th., Hagenbruch, O.: *Taschenbuch Mikroprozessortechnik*. Fachbuchverlag Leipzig, 2001.
- [2] Henning, P. A.; Vogelsang, H.:*Taschenbuch Programmiersprachen*. Fachbuchverlag Leipzig, 2007.
- [3] Slawig, Th.: *Programmiermethoden in der Mathematik*. Skript, TU Berlin, WS05/06.
- [4] Gessler, R.; Mahr, Th.: *Hardware-Software-Codesign*. Vieweg+Teuner, 2007.
- [5] Bollow, F.; Homann, M.; Köhn, K.-P: *C und C++ für Embedded Systems*. mitp, 2006.
- [6] INffORUM: *V-Modell - Entwicklungsstandard für IT-Systeme des Bundes*. http://www.infforum.de/themen/anwendungsentwicklung/thema_SE_v-modell.htm, 2011.