CT1 Zusammenfassung

Joël Plambeck – plambjoe@students.zhaw.ch – Version 0.1, 02.11.2020

Table of Contents

Dokumentation: Moodle

Computer Technik

Neumann Architecture	Hardware Components	Computers System & Executable

Main memory	central memory, via System-Bus, access to individual bytes, volatile
Secondary storage	long-term storage, via I/O, access to data blocks, non-volatile, slow but cheap

From C to executable

	Pre-Processor • Kopiert inhalt von #includes • Kopirt Makros (#define) in Quelltext **Compiler** • Übersetzt Computer unabhängigen C code in Assemblercode **Assembler** • Übersetzt in Maschineninstruktionen • Resultat: Binary File (nicht lesbar) **Linker** • Merge Object files • Löst Abhängigkeiten auf • Erzeugt executable

---

Cortex-M

Cortex-M Architecture	**Program Execution**	**Code structure**
Cortex-M Architecture	Program Execution	
Instruction Types	**Memory Allocation**	**C to Assembly**

Big endian

A muli-byte representation where the **MSByte** is at the lower address

Little endian

A muli-byte representation where the **LSByte** is at the lower address

Alignment

- Half-word aligned Variables aligned on even addresses
- Word aligned Variables aligned on addresses that are divisible by four

Object File Sections

Object File Sections	**Assembly Program Structure**	**Variables in Object sections**

Data Transfer

Arrays	Loading Literals	Literals variations
byte_array DCB 0xAA, 0xBB, 0xCC halfword_array DCW 0x0011, 0x2233		

Flags

Flag	Meaning	Action	Operands
Negative	MSB = 1	N = 1	signed
Zero	Result = 0	Z = 1	signed, unsigned
Carry	Carry	C = 1	unsigned
Overflow	Overflow	V = 1	signed

Arithmetic Operations

Bitwise operations	Shift / Rotate

Unsigned / Signed Integers

Addition / Subtraction

Unsigned

- **Addition**: C = 1 → Carry. Result too large for available bits
- **Subtraction**: C = 0 → Borrow. Result less than Zero (no negative numbers)

Signed

- **Addition**: potential **overflow** with equally signed operands
- **Subtraction**: potential **overlfow** with oppositely signed operands

Multiword Addidtion / Subtraction

Multi-Word Addition	Multi-Word Subtraction

Multiword addition **Multi-Word Addition**	Multiword addition **Multi-Word Subtraction**

Integer Casting

	Unsigned (Carry)	**Signed (Overflow)**
Extension	**0000** 1011 → 1011 **0000** 0011 → 0011	**1111** **1**011 → 1011 **0000** **0**011 → 0011
Truncation	Modulo Operation	possible change of sign

Loops

if	**do-while**	**while**	**switch**

Branches

- Type
  - **Unconditional**: jump always
  - **Conditional**: jump only if condition is met
- Address hand-over
  - **Direct**: target address part of instruction
  - **Indirect**: target address in register
- Address of target
  - **Relative**: target address relative to PC
  - **Absolute**: absolute target address

Compare and Test

CMP
- SUBS without storing result but setting flags

TST
- AND without storing result but setting flags

Unconditional

Symbol	**Properties**
B	direct, relative

Symbol	Properties
BX	indirect, absolute

Conditional

Conditional branch limit	**Limited range of -256..254 Bytes** for label/pointer of conditional branch

Flags

Symbol	Condition	Flag
BEQ	Equal	Z = 1
BNE	Not equal	Z = 0
BCS	Carry set	C = 1
BCC	Carry clear	C = 0
BMI	Negative	N = 1
BPL	Positive or Zero	N = 0
BVS	Overflow	V = 1
BVC	No overflow	V = 0

Unsigned

Symbol	Condition (Unsigned)	Flag
BEQ	Equal	Z = 1
BNE	Not equal	Z = 0
BHS (= BCS)	>= greater than or equal	C = 1
BLO (= BCC)	< less than	C = 0
BHI	> greater than	C = 1 and Z = 0
BLS	□ less than or equal	C = 0 or Z = 1

Signed

Symbol	Condition (Signed)	Flag
BEQ	Equal	Z = 1
BNE	Not equal	Z = 0
BMI	Negative	N = 1
BPL	Positive or Zero	N = 0
BVS	Overflow	V = 1
BVC	No overflow	V = 0

Symbol	Condition (Signed)	Flag
BGE	>= greater than or equal	N == V
BLT	< less than	N != V
BGT	> greater than	Z == 0 and N == V
BLE	□ less than or equal	Z == 1 or N != V

Parameterübergabe

	Register	Globales Memory	Stack
Effizient	++	—	/
Reentry	/	—	++

Subroutine

Caller	Callee

Calling Assembly Subroutine from C

Assembly Subroutine from C

Stack

- ONLY 32bit (Word)
- Pushing and Poping of halfword and bytes not possible
- SP % 4 = 0 → word aligned
- Stack-limit < SP < Stack-base

Stack Frame

Linking

Exceptions

Exceptional Control Flow

Polling	Interrupt-Driven I/O
Reading of status registers in loop Exception Polling **Advantages** • Simle and straightforward • Implicit synchronization • Deterministic • No additional interrupt logic required **Disadvantages** • Busy wait → wastes CPU time • Long reaction times • Reduzierter Durchsatz	Interrupting program execution when the error occurs Exception interrupt 1. Initializes peripherals 2. Execute other tasks 3. Peripherals signal when they require attention 4. Events interrupt program execution **Advantages** ◦ No busy wait → better use of CPU time ◦ short reaction times **Disadvantages** ◦ No synchronization ◦ difficult debugging

Exceptions Cortex-M3/M4

Interrupt sources: IRQ0 - IRQ239	System exceptions	Vector Table & NVIC
• Peripherals singal to CPU of event requiring attention • Can alternatively be generated by software request • Asynchronous to instruction execution	• Reset: Restart of processor • NMI: Non-maskable Interrupt: Can't be ingored • Faults: Eg. undefined instructions, analigned access, etc. • System Level Calls: OS calls	

Initialization	ISR Call	Exception States

Interrupts

Trigger hardware interrupt via Software	*Interrupt Active Status Registers*
Enable Registers	*Priority*

Pipeline

SEP Handout

Version 0.1
Last updated 2021-01-07 14:45:54 +0100