ARM Einführung

ARM = Advanced RISC Machines
RISC = Reduced Instructions Set Computing (Controller arbeiten mit reduziertem Befehlssatz)
CISC = Complex Instructions Set Computing
System on Chip → „Betriebssystem auf Chip“ (enthält z.B. A/D-Wandler, Ethernet, DSP Co-Prozessor)

Eingebettete Systeme

- System sieht nicht wie ein Computer aus, hat aber Computerfunktionen
- Rechner, der im technischen Gerät integriert ist
- Bsp. Automatisches Garagentor → wenige Funktionen (soll möglichst günstig sein)
- Umgebendes System (z.B. Display, Sensoren, Tasten)
- Zusammenhang: Sensoren → Hard/Software → Aktoren

Prozessorenvergleich

- Mikroprozessor: ALU + Register → keine Peripherie
- Mikrocontroller: Peripherie (A/D-Wandler, Timer)
- DSP: schnelle Verarbeitung von Audio/Videodaten, gleiche Abläufe in hoher Geschwindigkeit
- FPGA: Programmierung (VHDL) von digitaler Logik

System on Chip

- Vorteil: Zuverlässigkeit → Hersteller muss für alle Komponenten des Systems garantieren
- Inhalt: I/O-Zellen, High-Speed-Interfaces, Speicher, Standard-Zellen
- Peripherie: z.B. GSM (Global System for Mobile Communication), Grafikbearbeitung

Mikrocontroller

- Programm läuft in Endlosschleife → Interrupt-Controller zum Unterbrechen
- Speicher: ROM (Flash zum Programmieren) und RAM (Datenspeicher)

ARM Ltd. Geschäftsmodell

- Verkauft wird: Interlectual Poverty + Software + Entwicklungssysteme + Service
- Bezahlt werden die Lizenzgebühren zur Nutzung der ARM-Umgebung

Spezialisierung Halbleiter-Industrie

- Gate-Länge eines FETs legt die Strukturgröße fest
- Entwicklungskosten steigen mit Verkleinerung der Gates
- Aktuelle Strukturgröße liegt bei ca. 33 nm

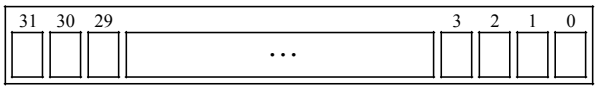
ARM 7 Grundlagen

- ARM 7 ist ein 32-Bit Controller
- 32-Bit entspricht 32 parallelen Datenleitungen
- 2^{32} verschiedene Adressen

  $2^{32} = 4,29 \cdot 10^9 \, Bit = 4,19 \cdot 10^6 \, kByte = 4056 MByte = 4 GByte$

  (Speicherbegrenzung auf 4 GB für 32-Bit Systeme)
- Load Store: Laden der Programmkomponenten → Berechnung → Speichern
- Rechnungen laufen in Registern ab

<u>32-Bit Register</u>



- Bits-Nr. von 0 – 31

<u>Befehlslänge</u>

- ARM-Modus (→ Vorlesungsinhalt) → Befehle mit 32 Bit Länge
- Thumb-Modus → Befehle mit 16 Bit Länge
- Befehle werden in der Regel in einem Taktzyklus bearbeitet (1 MHz = 10^6 Taktzyklen pro Sek.)

<u>Von Neumann Architektur</u>

- Ein Speicher → enthält sowohl ROM (Flash) als auch RAM (Daten)
- Nur eine Steuerleitung und nur ein Bus erforderlich

<u>Harward Architektur</u>

- Trennung von ROM & RAM (zwei Busse, zwei Steuerleitungen)
- Vorteil: DSP → eigene Steuerleitungen → schnellere Verarbeitung
- Nachteil: kein Ausweichen auf jeweils anderen Speicher möglich

<u>ARM 7 Aufbau</u>

- Modi: User-Mode, Interrupt-Mode, … (7 Betriebsmodi)
- 54 Befehle (ARM-Modus) + Bedingungen
- Befehlsabfolge: Fetch (Holen) → Decode (Dekodieren) → Execute (Ausführen)
- Pipelining zur Beschleunigung → Neuer Befehl wird geholt, während voriger dekodiert wird
- Programm-Counter → 32-Bit Register → enthält Adresse des Befehls, der als nächstes ausgeführt wird (4 Byte früher im Speicher: Decode, 8 Byte früher Execute)
- Thumb-Modus:

$$\left.\begin{array}{ll} PC & \boxed{Fetch} \\ PC-2 & \boxed{Decode} \\ PC-4 & \boxed{Execute} \end{array}\right\} 16-Bit\ Modus\,(Thumb)$$

- 16 Register im ARM-7 Prozessor:
    - R15: Program-Counter
    - R13: Stack-Pointer (Stapel-Zeiger) → enthält Inhalt des letzten gesicherten Registers
    - R14: Link-Register → enthält Sprungadresse bei z.B. Unterprogrammen

Assembler

- Befehle z.B. Addieren
  ADD R1, R2, R3 ; R1 = R2 + R3
- Compiler ist im Assemblerbereich der Assembler
- Quelltext wird vom Assembler in Maschinencode übersetzt → Object-Files
- Hinzufügen von Bibliotheken möglich → Linker zum Zusammenfügen
- Ablage im Speicher durch den Locator → korrekte Positionen
- Assembler-Directive (z.B. AREA ARMex, CODE, READONLY, „Name“) → Locator wird auf Lese-Speicher gestellt (READONLY)
- Sprungbefehl → Label ohne Leerzeichen
- Code mit mindestens einem Leerzeichen, kein Abschluss eines Befehls (Vgl. C → Semikolon)
- Abschluss des Programms mit END

<u>Kommentieren in Assembler</u>

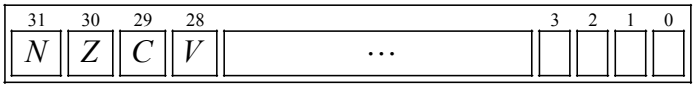
- Per Semikolon (Befehl ; Kommentar)
- Für gesamte Blöcke (/* Kommentar */)
- Zeilenweise (Befehl // Kommentar)

<u>ARM-7 Befehle</u>

- Integer (kein Floating-Point möglich) → ADD, SUB, MUL
- Logik-Befehle
- Retten von Registerinhalten beim Aufruf von Unterprogrammen → Stack-Befehle
- Swap-Befehle → bei vollem Speicher → Auslagerung auf z.B. Festplatte

<u>Current Program Status Register</u>

- Bits 0-4: Programmmodi
- Bit 5: Thumbmodus → 0/1 → 1 = Thumb
- Bits 6-7: Umschaltung IRQ / FIQ
- Bits 8-27: Reservierter Bereich (Reserve für z.B. zukünftige Anwendungen)
- Bits 28-31: Flags (N, Z, C, V)



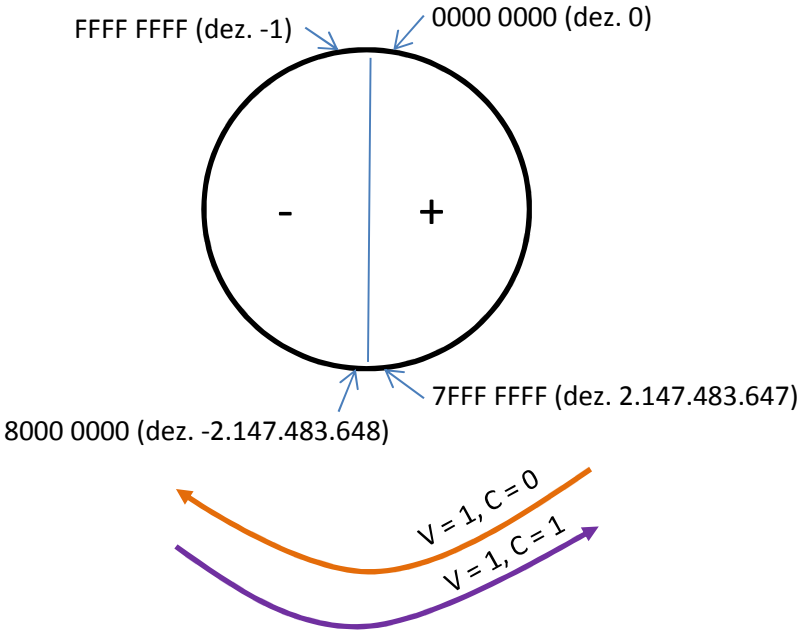
- Negative Flag (Bit 31) → bei negativen Ergebnissen → N = 1
- Zero-Flag (Bit 30) → Ergebnis ist Null → Z = 1
- Carry-Flag (Bit 29) → 33-Stelle im Ergebnis benötigt → C = 1
- Overflow-Flag (Bit 28)

Overflow-Flag

Durch den auf 32 Bit beschränkten Zahlenbereich kann es vorkommen, dass

- Bei einer Addition von zwei positiven Zahlen ein negatives Ergebnis entsteht
- Bei der Addition von zwei negativen Zahlen ein positives Ergebnis entsteht

Zahlenkreis 32 Bit:



Anmerkung:
Addition → im Uhrzeigersinn

Subtraktion → gegen den Uhrzeigersinn

Beispiele (mit 4-Bit Bereich):

- Addition von zwei positiven Zahlen

$$\begin{array}{rcccc} & 0 & 1 & 0 & 0 \\ + & 0 & {}_{1}1 & 1 & 0 \\ \hline C=0 & 1 & 0 & 1 & 0 \end{array}$$

- Prozessor analysiert Carry Flag (hier 0) und Übertrag auf letzte Stelle (hier 1) und bildet Antivalenz (XOR)
- Unterschiedliche Werte setzen V = 1
- Subtraktion von zwei negativen Zahlen

$$\left.\begin{array}{rcccc} & 1 & 0 & 0 & 1 \\ + & {}_{1}1 & {}_{0}0 & 0 & {}_{1}1 \\ \hline C=1 & 0 & 0 & 1 & 0 \end{array}\right\} C=1,\ \text{Übertrag} = 0 \Rightarrow V=1$$