Rechnerorganisation
Sommersemester 2022 – 7. Vorlesung

Prof. Stefan Roth, Ph.D.

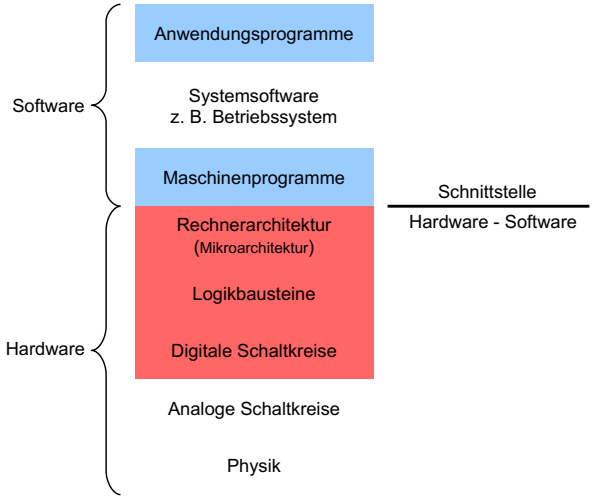
Technische Universität Darmstadt

12. Juni 2023

Inhalt

Mikroarchitekturen von Rechnersystemen

Schichtenmodell eines Computers



Komponenten eines Rechnersystems

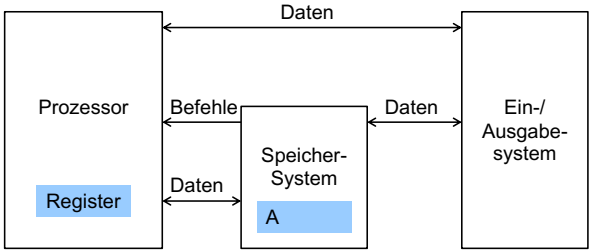


Abbildung: Komponenten eines Rechnersystems (verfeinerte Darstellung)

Anmerkung: Die Komponenten eines Rechnersystems müssen in der Regel eine gemeinsame Zeitbasis haben, damit das Zusammenspiel funktioniert. Diese Zeitbasis nennt man auch Takt/Taktsignal/Systemtakt.

Mikroarchitektur [HH16, S. 385 - 484]

- Einführung in die Mikroarchitektur
- Eintakt-Prozessor
- Mehrtakt-Prozessor
- Pipeline-Prozessor
- Analyse der Rechenleistung
- Ausnahmebehandlung
- Weiterführende Themen

Einführung in die Mikroarchitektur

Mikroarchitektur

- Hardware-Implementierung einer Architektur

Prozessor/Rechnersystem

- **Datenpfad**: verbindet funktionale Blöcke
- **Kontrollpfad**: Steuersignale/Steuerwerk

Mikroarchitektur

- Mehrere Implementierungen für eine Architektur (hier ARM®) möglich
- Unterscheidung ist der Takt bzw. die Art der Befehlsausführung
- Wie wird ein Befehl[1], z. B. **add**, ausgeführt?

Eintakt-Implementierung[a]

[a] Im Folgenden wird der Begriff Eintakt-Prozessor verwendet.

- Jeder Befehl wird in einem Takt ausgeführt.

Mehrtakt-Implementierung

- Jeder Befehl wird in Teilschritte zerlegt.

Pipelined-Implementierung

- Jeder Befehl wird in Teilschritte zerlegt. Mehrere Teilschritte werden gleichzeitig (parallel) ausgeführt.

[1] Instruktion, Maschinenbefehl

Mikroarchitektur eines ARM-Prozessors

Aus Assembler-Programmierung bekannter Prozessor und bekannte Befehle

Untermenge des ARM® Befehlssatzes

- Speicherbefehle: `ldr`, `str`
- Arithmetische/logische Befehle: `add`, `sub`, `and`, `orr`
- Sprungbefehle: `b`

Beispiel aus letzter Vorlesung – Objekt-Programm Schleife.o

```
Schleife.o:     file format elf32-littlearm
Disassembly of section .text:
00000000 <main>:
   0: e3a00001  mov r0, #1
   4: e3a01000  mov r1, #0
00000008 <WHILE>:
   8: e3500c01  cmp r0, #256 ; 0x100
   c: 0a000002  beq 1c <DONE>
   [...]
```

- Assembler: mov r0, #1
- Maschinensprache: e3a0 0001
  1110 0011 1010 0000 0000 0000 0000 0001
- Dieser Maschinenbefehl enthält (kodiert) alle Informationen und Anweisungen, das Register r0 mit dem Wert 1 zu füllen.

Architekturzustand

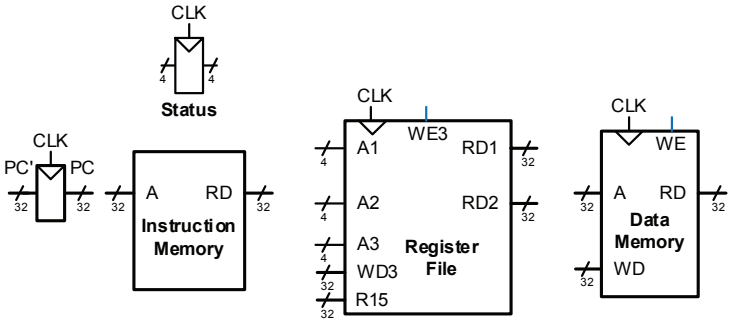
Architekturzustand: auf Ebene der Architektur sichtbare Daten

- Für den Programmierer zugänglich

Sichtbare Daten bestimmen den Zustand der Architektur

- PC (Program Counter, Befehlszähler)
- 16 Register
- Status-Flags
- Speicher

Elemente des Architekturzustands



Von-Neumann-Architektur, Harvard-Architektur, Speicher

Von-Neumann-Architektur

- gemeinsamer Speicher für Maschinenbefehle und Daten

Harvard-Architektur

- Befehlsspeicher und Datenspeicher sind getrennt

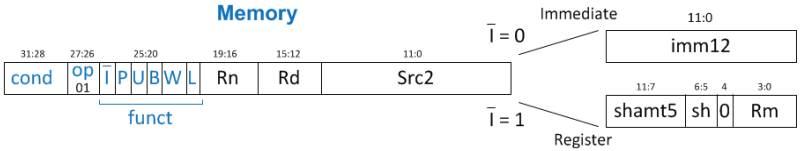
Verhalten des Speichers

- Die Speicher können, bezogen auf den Takt, asynchron gelesen werden
- Die Speicher werden synchron mit dem Takt geschrieben

Eintakt-Prozessor

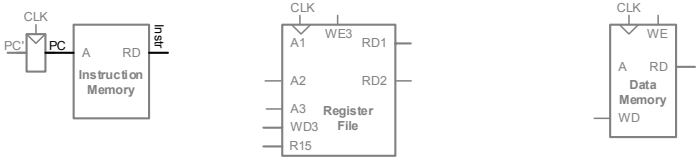
Eintakt-Prozessor, Vorgehensweise und Bitfelder eines Befehls

- Beginn der Entwicklung des Datenpfads mit dem Befehl `ldr`.
- Beispiel
  - `ldr r1, [r2, #5]`
  - `ldr Rd, [Rn, imm12]`



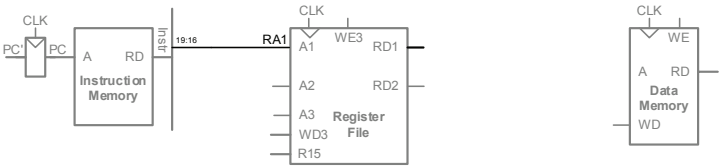
Eintakt-Prozessor, Datenpfad `ldr` Befehlsholphase

- 1. Befehl holen



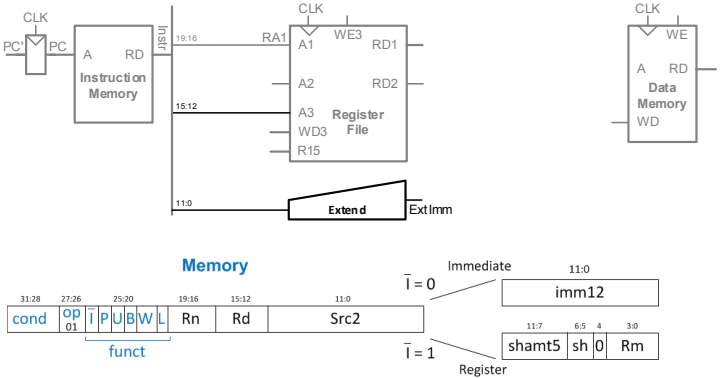
Eintakt-Prozessor, Datenpfad `ldr` Register lesen

- 2. Lesen der Quell-Operanden vom Registerfeld



Eintakt-Prozessor, Datenpfad `ldr` immediate

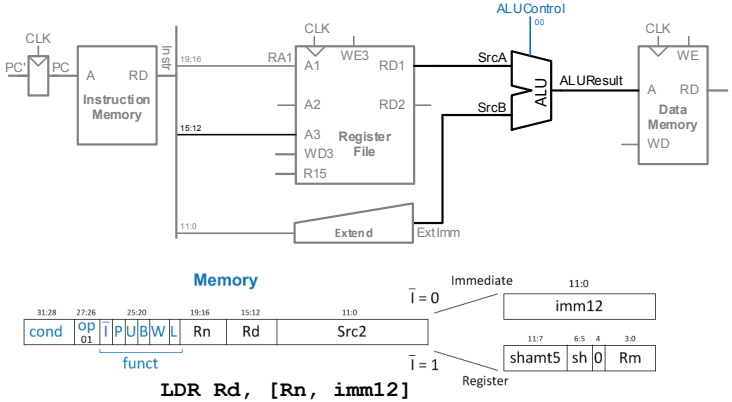
- 3. Erweiterung des immediates[2]. Der Offset ist als vorzeichenloser Wert (12 Bit) im Befehl gespeichert. **Extend** erweitert den Wert auf 32 Bit (mit Nullen).



[2] Direktwert

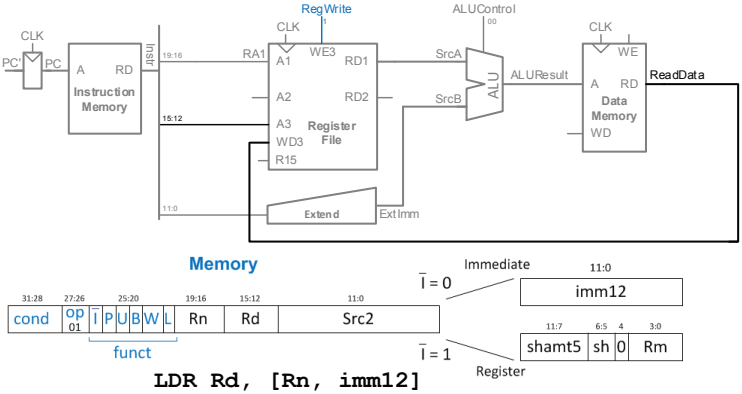
Eintakt-Prozessor, Datenpfad `ldr` Adressrechnung

- 4. Berechnung der Speicheradresse



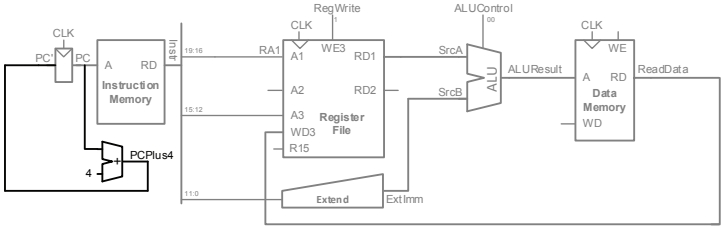
Eintakt-Prozessor, Datenpfad `ldr` Speicher lesen

- 5. Lesen der Daten aus dem Speicher und Schreiben in das Registerfeld



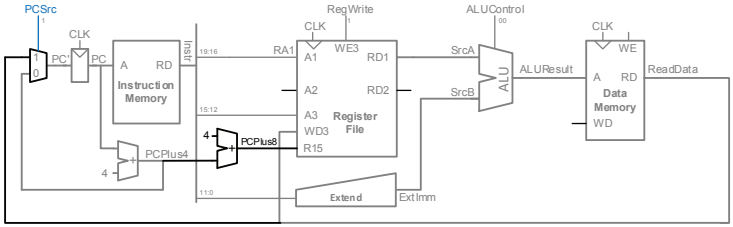
Eintakt-Prozessor, Datenpfad `ldr` PC erhöhen

- 6. Berechnung der Adresse des nächsten Befehls



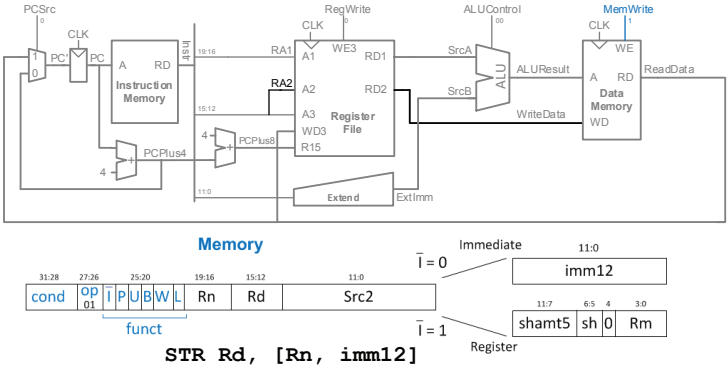
Eintakt-Prozessor, Datenpfad `ldr` Zugriff auf PC

- PC kann Quelle oder Ziel bei Instruktionen sein
  - Quelle: r15 muss im Registerfeld verfügbar sein. PC wird gelesen als aktueller PC + 8 (Zitat: aus historischen Gründen)
  - Ziel: Es muss möglich sein, PC als als Zielregister zu verwenden.



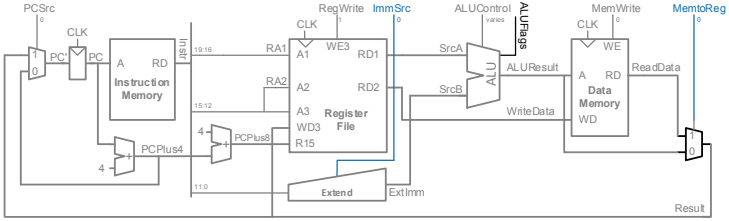
Eintakt-Prozessor, Datenpfad `str`

- Erweiterungen des Datenpfads zur Realisierung des Befehls `str`
- Schreibe Datum vom Registerfeld in den Datenspeicher

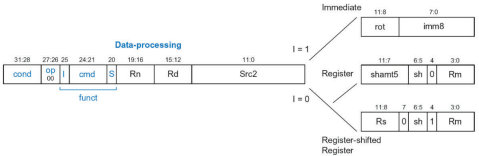


Eintakt-Prozessor, Datenpfad für arithmetische/logische Befehle

- mit immediate Src2; Steuersignal **ImmSrc**: wenn 0, Erweiterung um 24 Bit (für arithmetische/logische Befehle); wenn 1, Erweiterung um 20 Bit (für ldr/str)
- Schreibe ALUResult in Registerfeld (Register Rd)

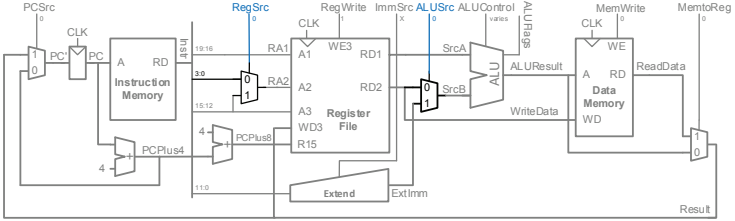


ADD Rd, Rn, imm8

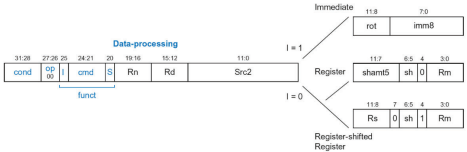


Eintakt-Prozessor, Datenpfad für arithmetische/logische Befehle

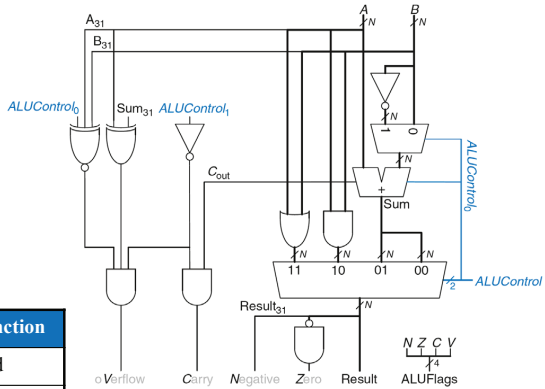
- mit Register Src2
- Schreibe ALUResult in Registerfeld (Register Rd)



ADD Rd, Rn, Rm



Arithmetisch Logische Einheit (ALU)

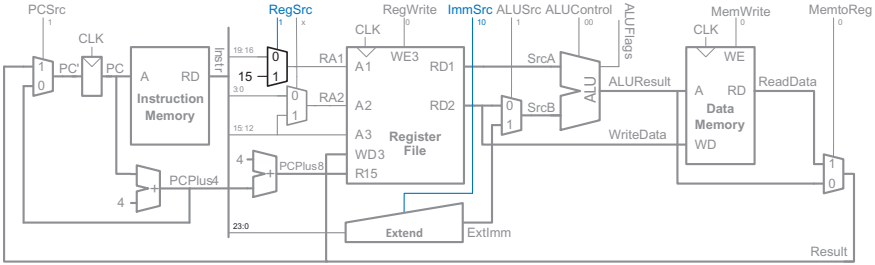


$\text{ALUControl}_{1:0}$	Function
00	Add
01	Subtract
10	AND
11	OR

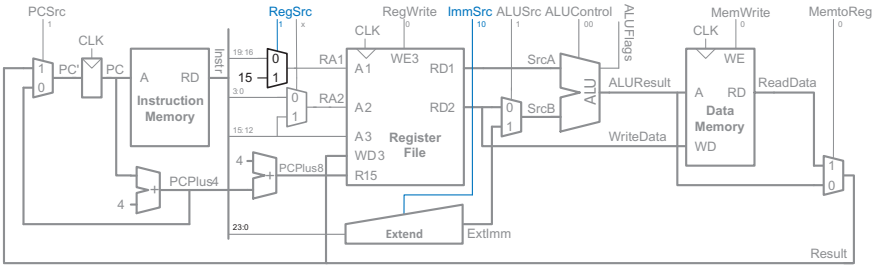
Flags must be stored in FFs if required by by instruction (e.g., CMP, TST, SUBS, ADDS, etc)

Eintakt-Prozessor, Sprungbefehl b

- Berechnen der Sprungadresse
- BTA = (ExtImm) + (PC + 8)
- ExtImm = Imm24 << 2 inkl. Vorzeichenerweiterung



Eintakt-Prozessor, ExtImm I

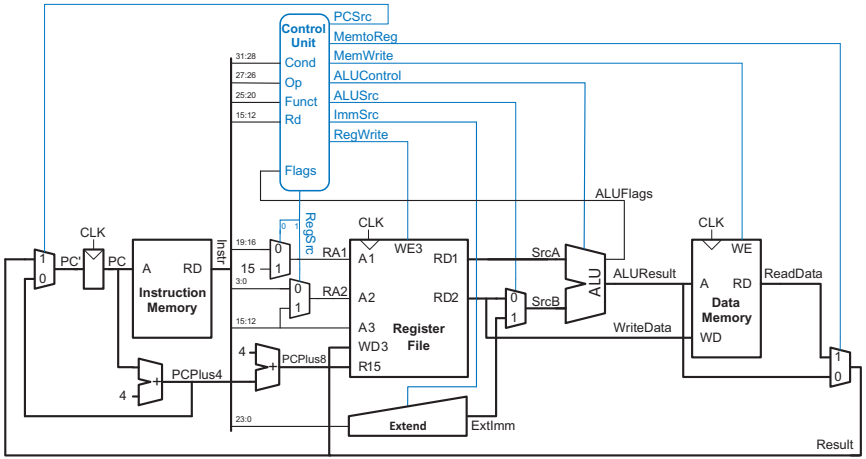


$\text{ImmSrc}_{1:0}$	ExtImm	Beschreibung
00	$\{24'b0, \text{Instr}_{7:0}\}$	Zero-extended `imm8` (z. B. add)
01	$\{20'b0, \text{Instr}_{11:0}\}$	Zero-extended `imm12` (z. B. ldr/str)
10	$\{6\{\text{Instr}_{23}\}, \text{Instr}_{23:0}00\}$	Sign-extended `imm24` multiplied by 4 (z. B. b)

Eintakt-Prozessor, ExtImm II

- Funktionserklärung der Einheit ExtImm
- Für die bisher vorgestellten Befehle sind unterschiedliche Funktionen notwendig
  - Die arithmetischen/logischen Befehle erlauben einen 8 Bit Wert als Direkwert und müssen entsprechend um 24 Bit erweitert werden (vgl. Zeile 1 in Tabelle).
  - Im Befehl `ldr` besteht die Möglichkeit, einen Direktwert der Breite 12 Bit zu speichern. Dieser Wert muss für die Adressrechnung auf 32 Bit erweitert werden. Es ist also notwendig, dass der Wert mit 20 Nullen zu einem 32 Bit Wert konkateniert wird (vgl. Zeile 2 in Tabelle).
  - Im Befehl b wird das Sprungziel als 24 Bit Adresse angegeben. Es ist eine vorzeichenrichtige Erweiterung vorzunehmen (vgl. Zeile 3 in Tabelle). Außerdem wird der Wert mit vier multipliziert (Linksshift um zwei Stellen).

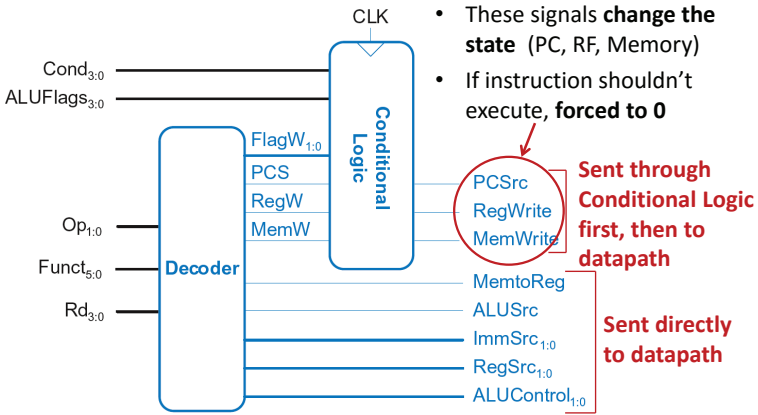
Eintakt-Prozessor, Datenpfad und Kontrolleinheit



Eintakt-Prozessor, Learning Nugget

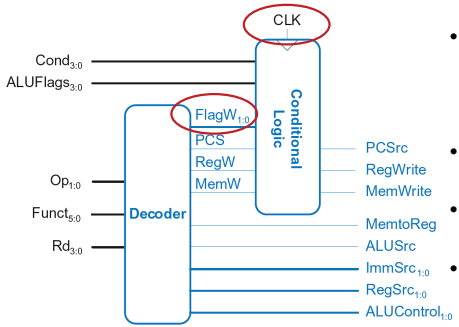
- Learning Nugget 05 – Eintakt-Prozessor

Eintakt-Prozessor, Kontrolleinheit



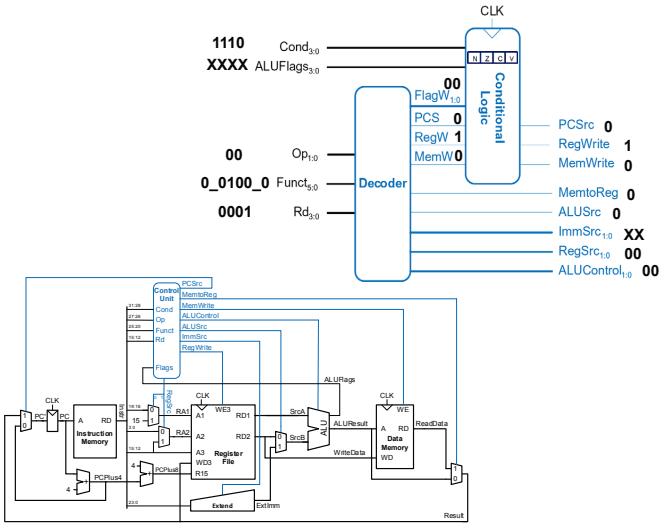
- These signals **change the state** (PC, RF, Memory)
- If instruction shouldn't execute, **forced to 0**

Eintakt-Prozessor, Kontrolleinheit

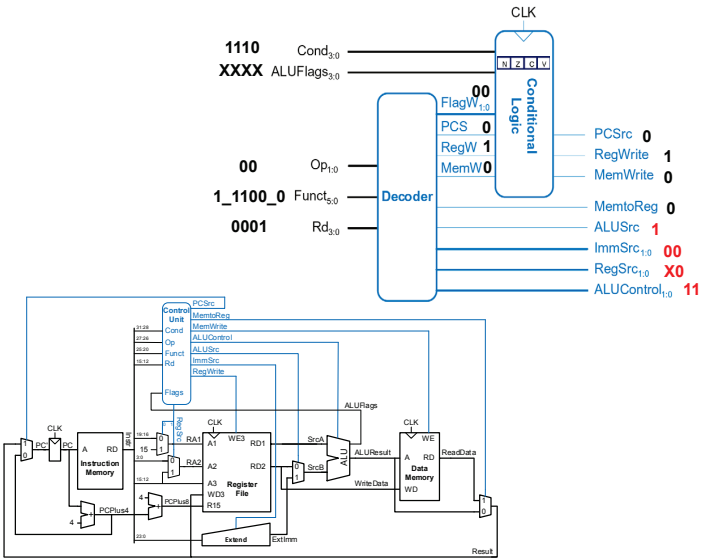


- ***FlagW*$_{1:0}$:** Flag Write signal, asserted when *ALUFlags* should be saved (i.e., on instruction with S=1)
- `ADD`, `SUB` update all flags (**NZCV**)
- `AND`, `ORR` only update **NZ** flags
- So, two bits needed:
  ***FlagW*$_1$** = 1: *NZ* saved (*ALUFlags*$_{3:2}$ saved)
  ***FlagW*$_0$** = 1: *CV* saved (*ALUFlags*$_{1:0}$ saved)

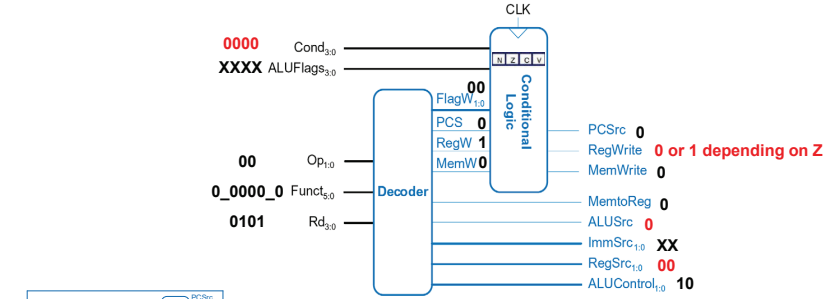
Eintakt-Prozessor, Beispiel `add r1,r2,r3`



Eintakt-Prozessor, Beispiel `orr r1,r2,#f0`



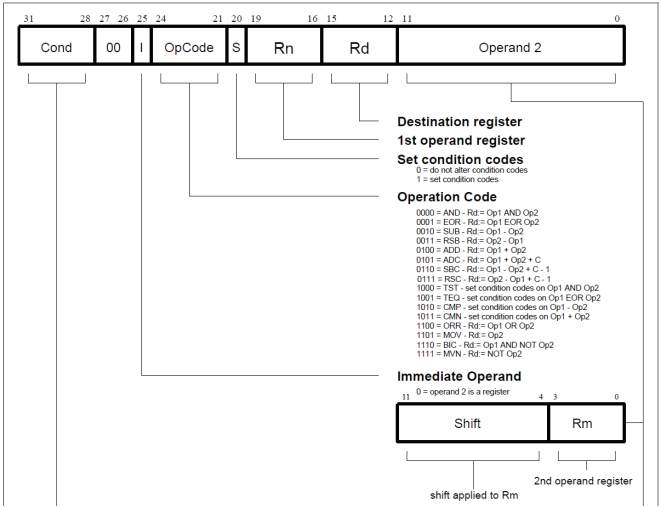
Eintakt-Prozessor, Beispiel `andeq r5,r6,r7`



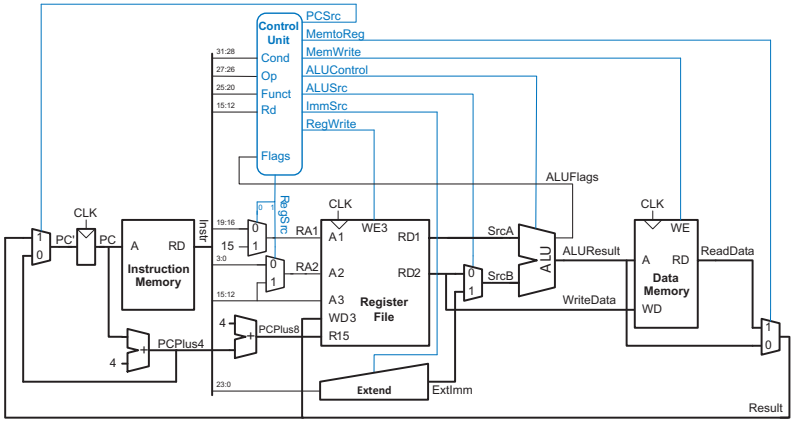
ARM® Instruction Set, Condition Codes der Befehle

Code	Suffix	Flags	Bedeutung
0000	EQ	Z gesetzt	Gleich
0001	NE	Z ungesetzt	Nicht gleich
0010	CS oder HS	C gesetzt	>= für unsigned
0011	CC oder LO	C ungesetzt	< für unsigned
0100	MI	N gesetzt	Negative
0101	PL	N ungesetzt	Positiv oder Null
0110	VS	V gesetzt	Overflow
0111	VC	V ungesetzt	Kein Overflow
1000	HI	C gesetzt und Z ungesetzt	> für unsigned
1001	LS	C ungesetzt oder Z gesetzt	<= für unsigned
1010	GE	N und V sind gleich	>= für signed
1011	LT	N und V sind ungleich	< für signed
1100	GT	Z ungesetzt, N und V sind gleich	> für signed
1101	LE	Z gesetzt, N und V sind ungleich	<= für signed
1110	AL	werden ignoriert	immer

ARM® Instruction Set, arithmetische/logische Befehle

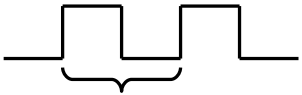


Eintakt-Prozessor, Datenpfad und Kontrolleinheit



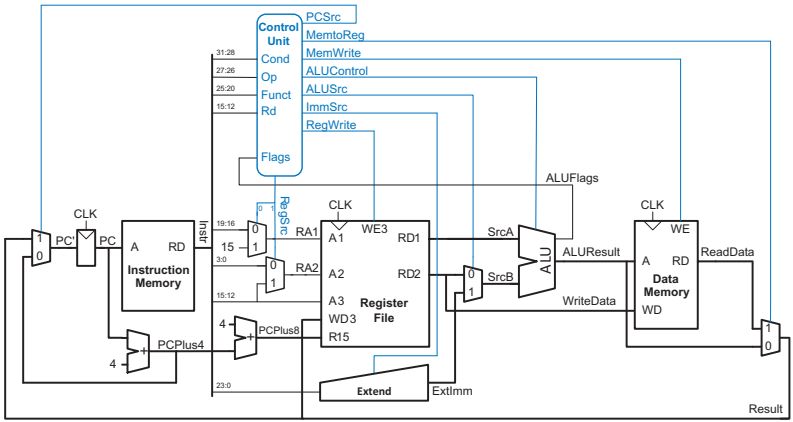
Eintakt-Prozessor, Diskussion

- Eintakt-Prozessor: bearbeitet jeden Befehl in einem Takt
- drei Phasen der Befehlsausführung
  - Befehlsholphase
  - Befehlsdekodierung
  - Befehlsausführung
- Taktsignal: Periodendauer/Taktfrequenz ($f = \frac{1}{T}$)



- Die Länge des **Pfades**, der zur Ausführung eines Befehls durchlaufen wird, ergibt eine Ausführungszeit
- Der längste Pfad gibt dann die maximale Taktfrequenz vor

Eintakt-Prozessor, Beispiele für Pfade



Zusammenfassung und Ausblick

Zusammenfassung und Ausblick

Zusammenfassung

- Mikroarchitekturen von Rechnersystemen
- Eintakt-Prozessor, Entwicklung des Datenpfads

Ausblick

- Mikroarchitekturen von Rechnersystemen
- Mehrtakt-Prozessor

Lernkontrolle

- Ich habe die Terminologie im Kontext von Rechnerarchitekturen/Prozessorarchitekturen verstanden und kann diese richtig nutzen ☑
- Ich habe die Phasen der Befehlsverarbeitung verstanden und mir eingeprägt ☑
- Die Methodik, aus der Wirkung eines Maschinenbefehls einen Datenpfad zu konstruieren habe ich verstanden ☑
- Ich habe die Strukturierung einer Mikroarchitektur in **Datenpfad** und **Kontrollpfad** verstanden ☑
- Ich bin in der Lage, mir die Steuersignale aus der Funktion der auszuführenden Befehle herzuleiten ☑
- ...

Literatur

Literatur

[BO10] Bryant, Randal E. und David R. O´Hallaron: *Computer Systems - A Programmer´s Perspective*.
Prentice Hall, 2010.

[HH16] Harris, David Money und Sarah L. Harris: *Digital Design and Computer Architecture, ARM® Edition*.
Morgan Kaufmann, 2016.