Rechnerorganisation

Sommersemester 2023

Prof. Stefan Roth, Ph.D.

TECHNISCHE UNIVERSITÄT DARMSTADT

9. Aufgabenblatt mit Lösungsvorschlag

26.06.2023
Mikroarchitekturen, Eintakt-Prozessor

Aufgabe 1: Theoriefragen

a) Welcher Teil des Prozessors bestimmt, welche Rechenoperation die ALU ausführt?

Lösungsvorschlag:

Die Steuereinheit (Control Unit) bestimmt ALUControl. Abgeleitet wird das aus dem Bitfeld funct.

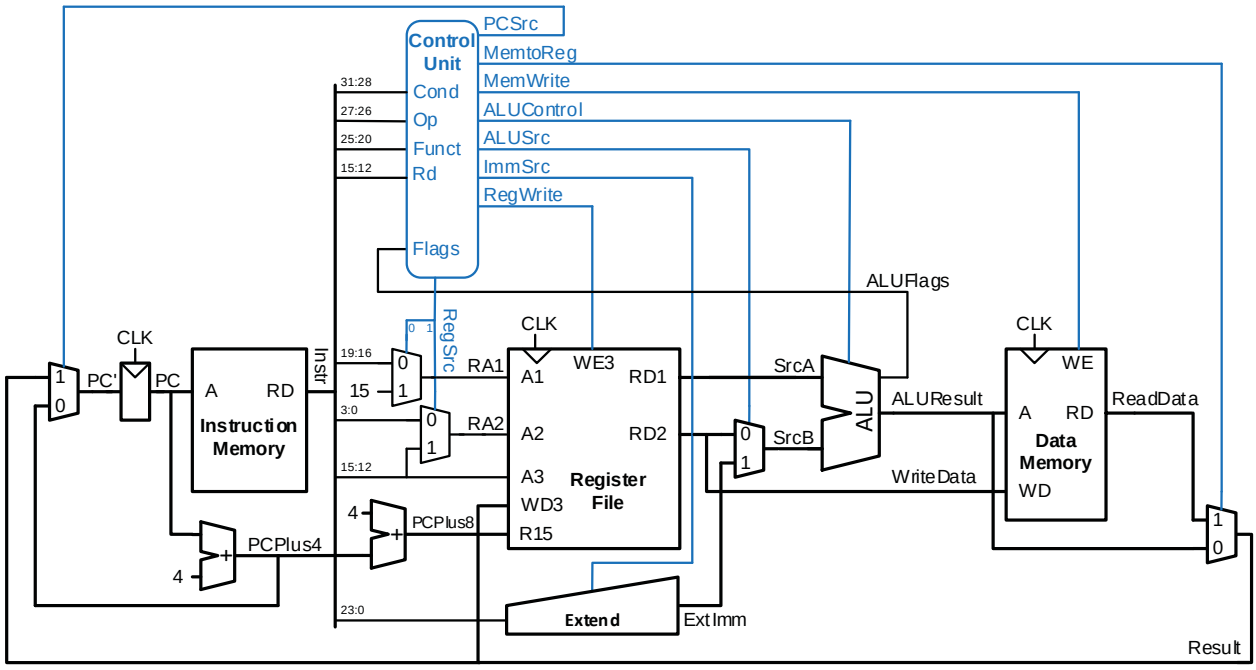
b) Durch was wird die Taktfrequenz beim Eintakt-Prozessor bestimmt?

Lösungsvorschlag:

Die langsamste Instruktion bestimmt die Taktfrequenz. Die Taktfrequenz ergibt sich aus den Verzögerungszeiten im Datenpfad/in der Steuereinheit.

Aufgabe 2: Steuersignale ausgewählter Befehle

In der Vorlesung haben Sie den folgenden Eintakt-Prozessor kennengelernt.



Geben Sie für die folgenden Befehle die Belegung der Eingangs- und Ausgangssignale der `Control Unit` an.

- str r11, [r5,#8]

- bne there
- cmp r1, r2
- subs r3, r1, r2

Lösungsvorschlag:

str r11, [r5,#8]

Die Analyse des Befehls bzw. des Bitfeldes ist der erste Schritt, mit dem man die Eingangssignale in die `Control Unit` ermittelt. Dazu wird der Befehl in ein übersetzbares Assemblerprogramm eingefügt.[1]

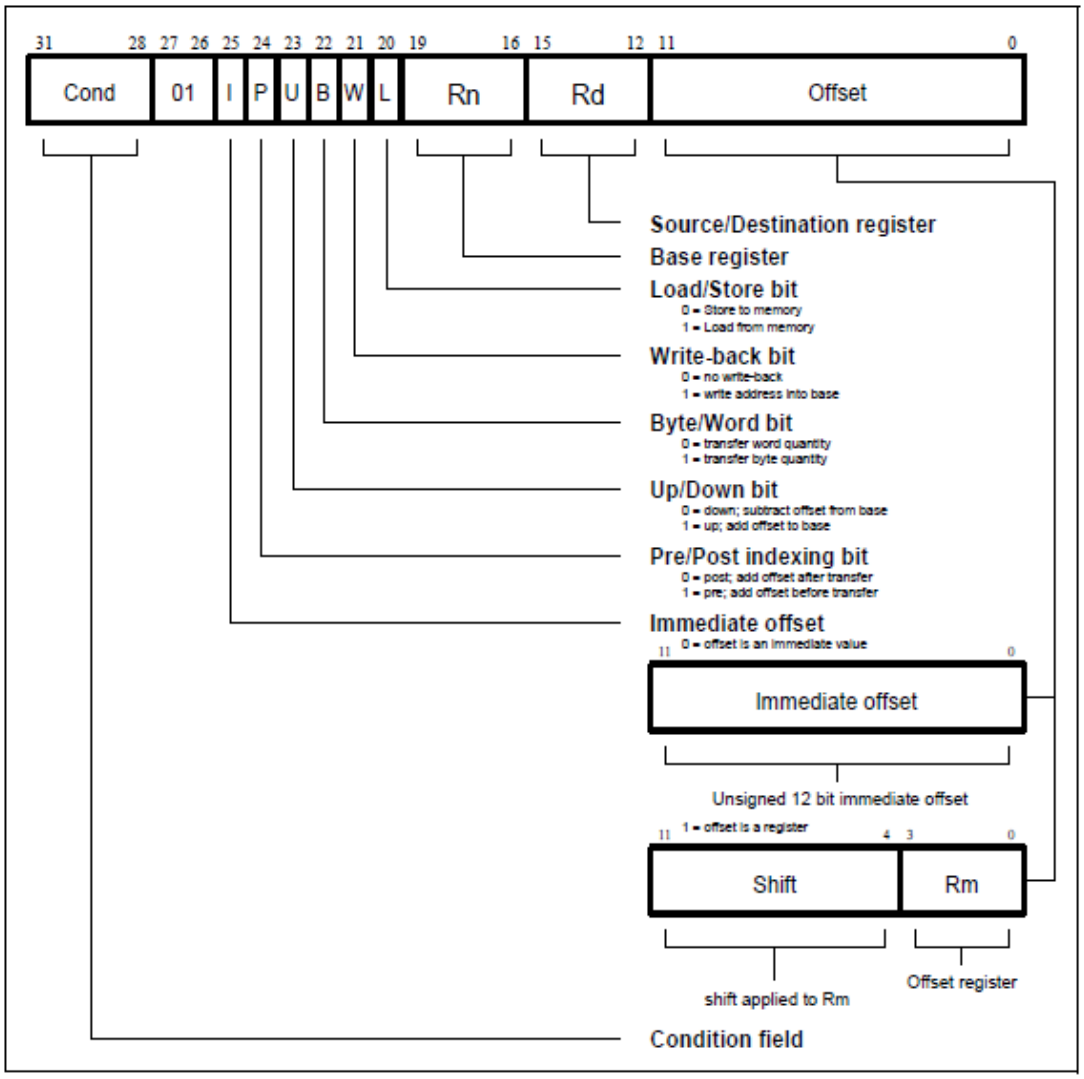
Object Dump:

```
00000000 <main>:
0: e585b008 str fp, [r5, 8]
4: e12fff1e bx lr
```

Der Store-Befehl ist hexadezimal codiert 0x e585b008. Die Umrechnung in das Dualsystem führt zu folgender Bitfolge:
`1110 01 0 1 1 0 0 0 0101 1011 000000001000`

Mittels Abschnitt 4.2 und 4.5 des ARM Instruction Set Handbuchs lassen sich die Bitfelder analysieren. Auf Seite 4-26 findet sich folgende Abbildung.

[1] Achtung: Die Ausführung des Programms führt zu einem Speicherzugriffsfehler. Warum das so ist, sollte nach kurzer Überlegung klar sein.

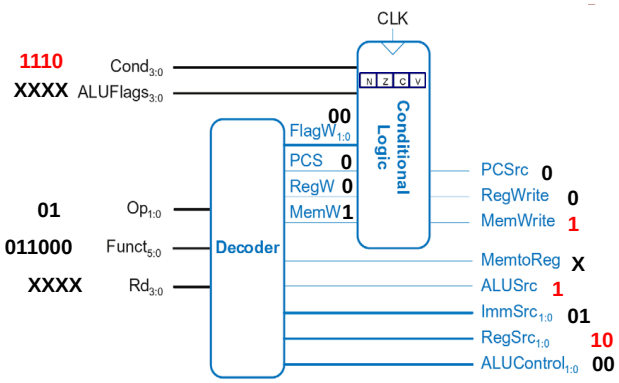


Damit ergibt sich für die Bitfelder folgende Werte. Hier sieht man auch, dass das Framepointer-Register (kurz fp) im Registersatz durch r11 abgebildet wird.

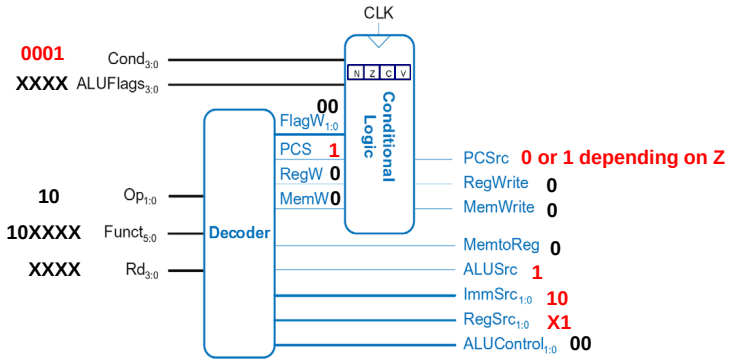
Bit(s)	Wert	Kommentar
31:28	1110	CC always
27:26	01	Opcode LDR/STR
25	0	0 = offset is an immediate value
24	1	1 = add offset before transfer
23	1	1 = add offset to base
22	0	0 = transfer word quantity
21	0	0 = no write back
20	0	0 = Store to memory
19:16	0101	Basisregister (r5)
15:12	1011	Quell-Register (r11 oder fp)
11:0	000000001000	Offset als Direktwert (8)

Mit diesen Informationen und dem Datenpfad des Befehls ergeben sich die Eingaben und die Ausgaben der `Control Unit`. Dabei sind die Bits von 25:20 zusammengefasst und werden als $\texttt{Funct}_{5:0}$ in den Decoder gegeben. Das $\texttt{Rd}_{3:0}$ für die Ausführung des Befehls nicht relevant ist, wird dadurch gekennzeichnet, dass ein X bzw. XXXX angegeben ist. Die Steuersignale für die Komponenten ergeben sich wie folgt (vgl. Datenpfad, der bei Ausführung von `str` benutzt wird).

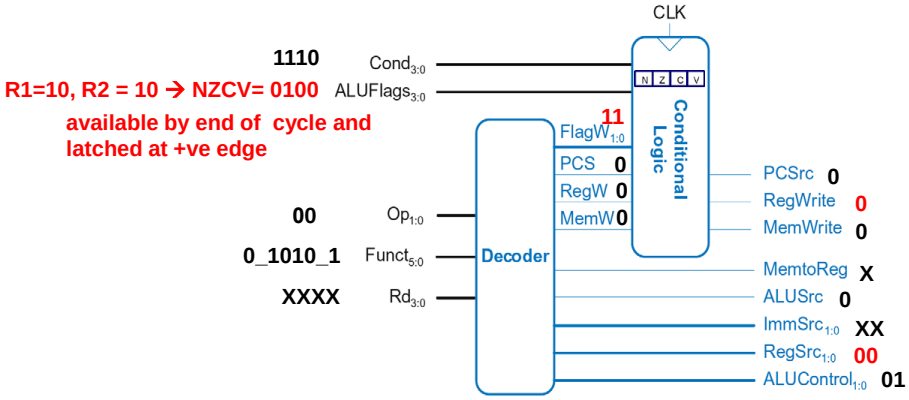
- PCSrc = 0 ⇒ der Multiplexer am Eingang des Programmzählers bekommt die nächste Adresse
- RegWrite = 0 ⇒ es wird nichts in das Registerfile geschrieben
- MemWrite = 1 ⇒ der Befehl schreibt den Inhalt des Register r11 in den Speicher; folglich muss der Speicher auf Schreiben geschaltet werden.
- MemtoReg = X ⇒ X hat die Semantik von don´t care. Es ist also egal, ob es 0 oder 1 ist.
- ALUSrc = 1 ⇒ da ein Direktwert (nämlich der Offset) mit der Basisadresse r5 addiert wird, muss der Multiplexer entsprechend geschaltet werden.
- ImmSrc = 01 ⇒ dieses Signal steuert, was mit dem Direktwert vor der Addition passiert (vgl. Vorlesung 7). Dem 12 Bitwert aus dem Befehl werden 20 Nullen vorangestellt.
- RegSrc = 10 ⇒ Schaltet den Multiplexer vor dem Registeradresseingang A2 entsprechend, dass das Quell-Register an das Registerfeld angelegt wird.
- AluControl = 00 ⇒ vgl. Vorlesung 7. Die ALU addiert den Inhalt von r5 und den Offset 8.



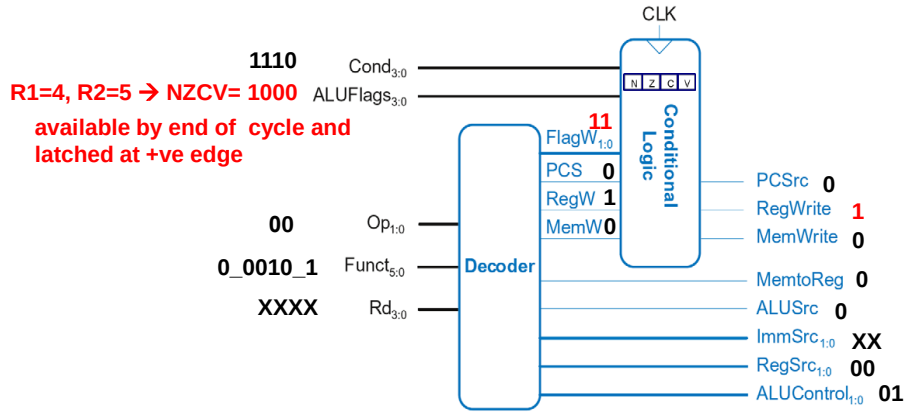
bne there



cmp r1, r2



subs r3, r1, r2



Aufgabe 3: Erweiterung des Eintakt-Prozessors

Die arithmetisch/ logischen Befehle erlauben es, ein Shift im Befehl auszuführen.
Ein Besipiel dafür ist der Befehl `add r7, r2, r12, lsr #5`.

Betrachtet wird das folgende Assemblerprogramm.

```
/* -- analysis.s */
/* Kommentar  */
.global main /* Einsprungpunkt Hauptprogramm */

main:           /* Hauptprogramm */
    mov r1, #42  /* Schreibe eine 42 in das Register r1 */
    mov r2, #4   /* Schreibe eine 5 in das Register r2 */
                 /* Addiere die Register r1 und r2 */
```

```
            /* r2 / 2 durch Rechtsshift */
add r0,r1,r2,lsr #1
bx lr       /* Springe zurueck zum aufrufenden Programm */
```

1. Assemblieren und Linken Sie das Programm. Schauen Sie sich nun den Object Dump an. Suchen Sie den Additionsbefehl und analysieren Sie die Belegung des Bitfeldes. Nutzen Sie dazu das ARM Instruction Set[2] Handbuch. Insbesondere das Studium von Abschnitt 4.2, 4.5 und 4.5.2 sollten hilfreich sein.

Lösungsvorschlag:

Object Dump:

```
00010408 <main>:
10408: e3a0102a mov r1, 42 ; 0x2a
1040c: e3a02004 mov r2, 4
10410: e08100a2 add r0, r1, r2, lsr 1
10414: e12fff1e bx lr
```

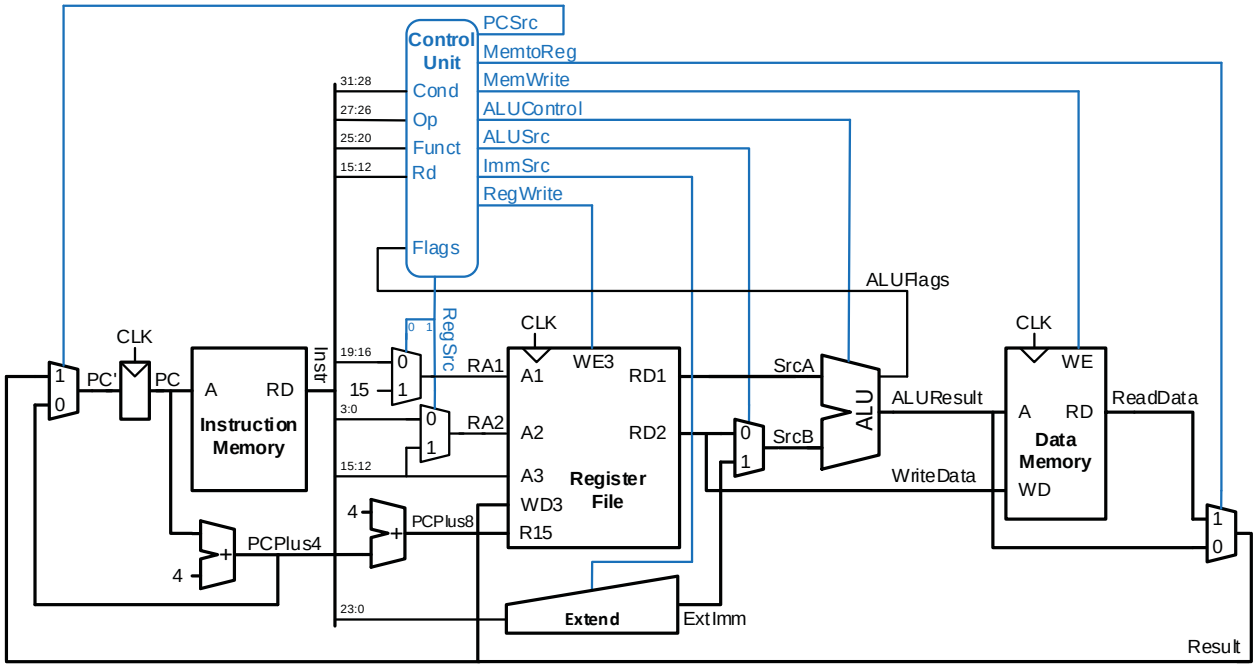
Der Additionsbefehl (mit Shift) ist hexadezimal codiert 0x e08100a2. Die Umrechnung in das Dualsystem führt zu folgender Bitfolge: `1110 00 0 0100 0 0001 0000 0000 1010 0010`

Mittels Abschnitt 4.2, 4.5 und 4.5.2 des ARM Instruction Set Handbuchs lassen sich die Bitfelder analysieren.

Bit(s)	Wert	Kommentar
31:28	1110	CC always
27:26	00	Opcode Data Processing
25	0	0 = operand 2 is a register
24:21	0100	Operation Code (für add)
20	0	0 = do not alter condition codes
19:16	0001	Quellregister (r1)
15:12	0000	Zielregister (r0)
11:7	0000 1	Shiftweite (=1)
6:5	01	logical right
4	0	Null
3:0	0010	2nd operand register (r2)

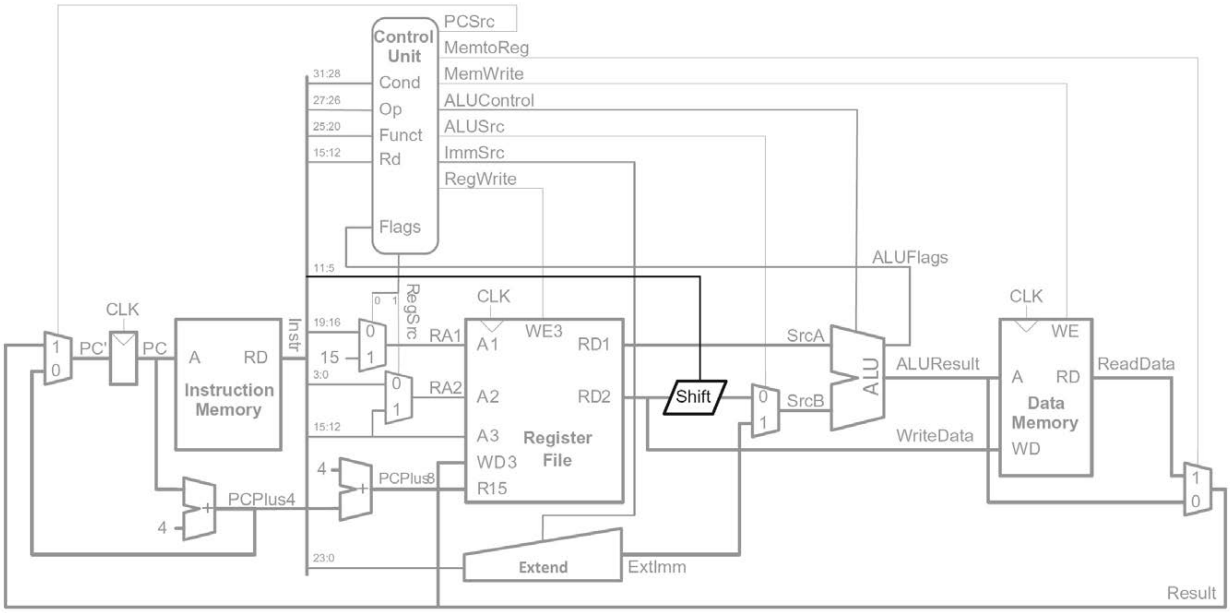
2. Überlegen Sie, ob der Datenpfad des Eintakt-Prozessors erweitert werden muss und führen Sie diese Erweiterung durch.

[2] https://moodle.tu-darmstadt.de/pluginfile.php/1809291/mod_folder/content/0/Material/arm-instructionset.pdf


Control Unit
Cond
Op
Funct
Rd
Flags
PCSrc
MemtoReg
MemWrite
ALUControl
ALUSrc
ImmSrc
RegWrite
ALUFlags
31:28
27:26
25:20
15:12
19:16
15
3:0
15:12
23:0
RegSrc
CLK
PC'
PC
A
RD
Instruction Memory
Instr
RA1
RA2
A1
A2
A3
WD3
R15
WE3
RD1
RD2
Register File
PCPlus8
PCPlus4
4
Extend
ExtImm
SrcA
SrcB
ALU
ALUResult
WriteData
WE
WD
Data Memory
ReadData
Result

Lösungsvorschlag:

Der Datenpfad muss wie folgt erweitert werden.



An der Control Unit sind keine Änderungen notwendig.