Anhang A: Prozessor

Dieser Anhang enthält die Dokumentation des Prozessors.

Architektur

Die CPU entspricht einer von Neumann Architektur. Strukturell setzt sich die CPU aus den folgenden Komponenten zusammen:

1. Registerwerk
2. Arithmetisch-Logische Einheit (ALU)
3. Steuerwerk

Eine strukturelle Darstellung der Einheiten einschließlich der Signal-, Entity- und Dateinamen ist in Abbildung 1 dargestellt.

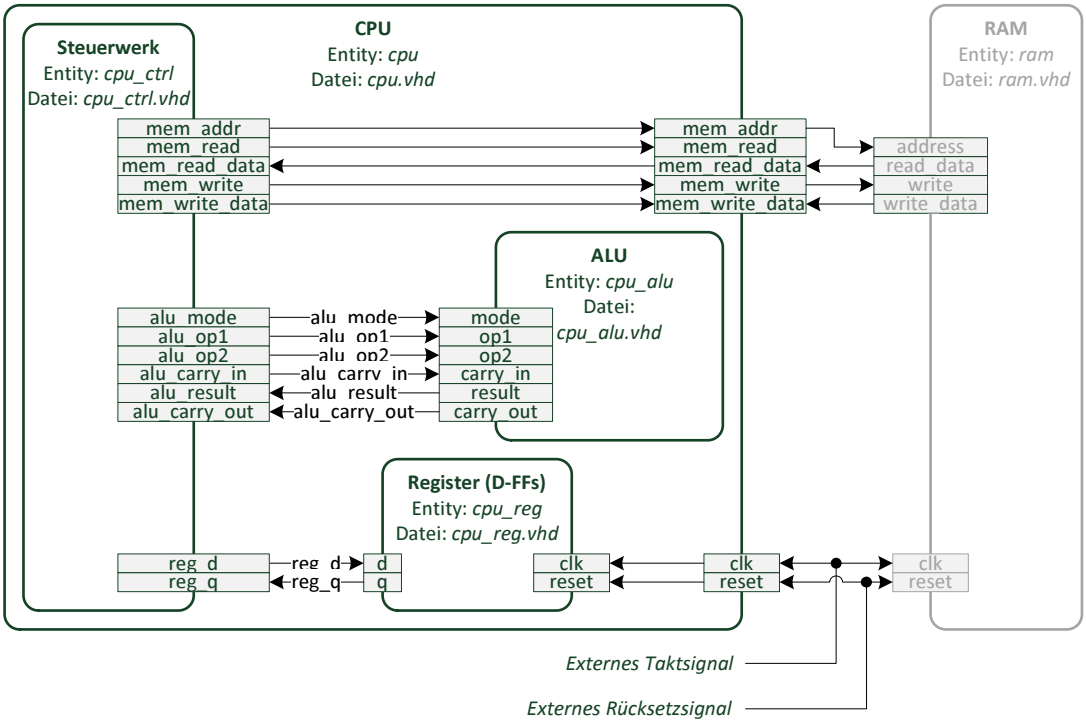

Abbildung 1: Strukturelle Unterteilung der CPU.

Der CPU verwendet einen externen Speicher (RAM) für Programm und Daten (ebenfalls in der Abbildung dargestellt).

Johannes Specht (specht@dc.uni-due.de)

Befehlsformat

Das Befehlsformat der CPU verwendet 16 Bit breite Befehle. Das Befehlsformat ist in Tabelle **1** dargestellt.

CTRL inst(15...12)	ALU inst(11...8)	Direkt-Operand/Adresse inst(7...0)

Befehlssatz

Mnemonic	Semantik des Befehls-operanden	CTRL	ALU	Hex.	Beschreibung
LOAD	abs. Adresse	MODE8	BYPASS	8Fxx	Lädt einen Wert aus dem Speicher in den Akkumulator.
LOAD	rel. Addresse	MODE11	BYPASS	BFxx	Lädt einen Wert aus dem Speicher in den Akkumulator.
LOAD	Direktoperand	MODE3	BYPASS	2Fxx	Lädt den Direktoperanden in Akkumulator.
STORE	abs. Adresse	MODE10	BYPASS	AFxx	Speichert den Wert des Akkumulators.
STORE	rel. Adresse	MODE13	BYPASS	DFxx	Speichert den Wert des Akkumulators.
PUSH	Register-Index	MODE15	BYPASS	FFxx	Legt den Wert eines bestimmten Registers auf dem Stack ab. Das Register wird im Befehlsoperanden angegeben: 0 : PC[1] 1 : ACC Sollte der Stack voll sein, so wird der neue Wert über den letzten geschrieben.
POP	Register-Index	MODE14	BYPASS	EFxx	Liest den obersten Wert des Stacks in ein bestimmtes Register. Das Register wird im Befehlsoperanden angegeben: 0 : PC 1 : ACC Sollte der Stack leer sein, so wird ein undefinierter Wert zurückgegeben.
JMP	abs. Adresse	MODE5	BYPASS	4Fxx	Springt zu einer Adresse.
JMP	rel. Adresse	MODE5	ADD	40xx	Springt zu einer Adresse.
JC	abs. Adresse	MODE6	BYPASS	5Fxx	Springt zu einer Adresse, falls das Carry-Flag (Übertrag) gesetzt ist.
JC	rel. Adresse	MODE6	ADD	50xx	Springt zu einer Adresse, falls das Carry-Flag (Übertrag) gesetzt ist.
JZ	abs. Adresse	MODE7	BYPASS	6Fxx	Springt zu einer Adresse, falls das Zero-Flag (Null-Flag) gesetzt ist.
JZ	rel. Adresse	MODE7	ADD	60xx	Springt zu einer Adresse, falls das Zero-Flag (Null-Flag) gesetzt ist.
NOP	-	MODE1	BYPASS	0Fxx	Führt einen Befehlszyklus ohne sonstige Auswirkung aus.
ADD	abs. Adresse	MODE9	ADD	90xx	Addiert einen Wert aus dem Speicher zu dem Akkumulator und aktualisiert das Carry-Flag.
ADD	rel. Adresse	MODE12	ADD	C0xx	Addiert einen Wert aus dem Speicher zu dem Akkumulator und aktualisiert das Carry-Flag.
ADD	Direktoperand	MODE2	ADD	10xx	Addiert einen Direktoperanden zu dem Akkumulator und aktualisiert das Carry-Flag.
ADC	abs. Adresse	MODE9	ADC	93xx	Addiert einen Wert aus dem Speicher und das (alte) Carry-Flag zu dem Akkumulator und aktualisiert das Carry-Flag.
ADC	rel. Adresse	MODE12	ADC	C3xx	Addiert einen Wert aus dem Speicher und das (alte) Carry-Flag zu dem Akkumulator und aktualisiert das Carry-Flag.
ADC	Direktoperand	MODE2	ADC	13xx	Addiert einen Direktoperanden und das (alte) Carry-Flag zu dem Akkumulator und aktualisiert das Carry-Flag.
SUB	abs. Adresse	MODE9	SUB	94xx	Subtrahiert einen Wert aus dem Speicher von dem Akkumulator und aktualisiert das Carry-Flag.
SUB	rel. Adresse	MODE12	SUB	C4xx	Subtrahiert einen Wert aus dem Speicher von dem Akkumulator und aktualisiert das Carry-Flag.
SUB	Direktoperand	MODE2	SUB	14xx	Subtrahiert einen Direktoperanden von dem Akkumulator und aktualisiert das Carry-Flag.
NEG	-	MODE4	NEG	35xx	Negiert (Zweierkomplement) den Akkumulator.

[1] Auf dem Stack wird die Adresse des PUSH-Befehls + 2 hinterlegt.

Mnemonic	Semantik des Befehls-operanden	CTRL	ALU	Hex.	Beschreibung
NOT	-	MODE4	NOR	37xx	Invertiert den Akkumulator.
AND	abs. Adresse	MODE8	AND	81xx	Führt eine UND-Verknüpfung zwischen einem Wert aus dem Speicher und dem Akkumulator durch.
AND	rel. Adresse	MODE11	AND	B1xx	Führt eine UND-Verknüpfung zwischen einem Wert aus dem Speicher und dem Akkumulator durch.
AND	Direktoperand	MODE2	AND	11xx	Führt eine UND-Verknüpfung zwischen einem Direktoperanden und dem Akkumulator durch.
OR	abs. Adresse	MODE8	OR	86xx	Führt eine OR-Verknüpfung zwischen einem Wert aus dem Speicher und dem Akkumulator durch.
OR	rel. Adresse	MODE11	OR	B6xx	Führt eine OR-Verknüpfung zwischen einem Wert aus dem Speicher und dem Akkumulator durch.
OR	Direktoperand	MODE2	OR	16xx	Führt eine OR-Verknüpfung zwischen einem Direktoperanden und dem Akkumulator durch.
NOR	abs. Adresse	MODE8	NOR	87xx	Führt eine NOR-Verknüpfung zwischen einem Wert aus dem Speicher und dem Akkumulator durch.
NOR	rel. Adresse	MODE11	NOR	B7xx	Führt eine NOR-Verknüpfung zwischen einem Wert aus dem Speicher und dem Akkumulator durch.
NOR	Direktoperand	MODE2	NOR	17xx	Führt eine NOR-Verknüpfung zwischen einem Direktoperanden und dem Akkumulator durch.
RCL	-	MODE2	RCL	1Bxx	Schiebt den Inhalt des Akkumulators um 1 Bit nach links. Das neue Carry-Flag wird auf den Wert des hinausgeschobenen Bits gesetzt, das hineingeschobene Bit hat den Wert des alten Carry-Flags.
RCR	-	MODE2	RCR	1Axx	Schiebt den Inhalt des Akkumulators um 1 Bit nach rechts. Das neue Carry-Flag wird auf den Wert des hinausgeschobenen Bits gesetzt, das hineingeschobene Bit hat den Wert des alten Carry-Flags.
SHL	-	MODE2	SHL	19xx	Schiebt den Inhalt des Akkumulators um 1 Bit nach links. Das Carry-Flag wird auf den Wert des hinausgeschobenen Bits gesetzt, das hineingeschobene Bit hat den Wert 0.
SHR	-	MODE2	SHR	18xx	Schiebt den Inhalt des Akkumulators um 1 Bit nach rechts. Das Carry-Flag wird auf den Wert des hinausgeschobenen Bits gesetzt, das hineingeschobene Bit hat den Wert 0.

Tabelle 1: Befehlssatz der CPU.

Johannes Specht (specht@dc.uni-due.de)

Registerwerk

Das Registerwerk der CPU lässt sich als Verbund zahlreicher paralleler Taktvorderflankengesteuerte D-Flipflops betrachten, wobei diese ein idealisiertes Zeitverhalten aufweisen. Die Flipflops bilden die einzelnen Register der CPU, die in nachfolgender Tabelle dargestellt sind.

Kürzel	Bitbreite	Beschreibung
PC	8	Bildet den Befehlszeiger des Prozessors (Program Counter)
ACC	16	Bildet den Akkumulator des Prozessors (Accumulator)
STATUS	2	Bildet das Statusregister des Prozessors und enthält die folgenden Status-Flags: - Carry (Übertrag) - Zero (ACC=0)
INST	16	Bildet das Befehlsregister des Prozessors (Instruction) und enthält das zuletzt geholte Befehlswort
MAR	8	Bildet das Speicheradressregister (Memory Address Register)
MDR	16	Bildet das Speicherdatenregister (Memory Data Register)
SP	8	Bildet den Stapelzeiger (Stack Pointer)

ALU

Bei der Arithmetisch-Logischen-Einheit (ALU) der CPU handelt es sich um ein Schaltnetz. Die nachfolgende Abbildung zeigt eine strukturelle Darstellung der ALU.

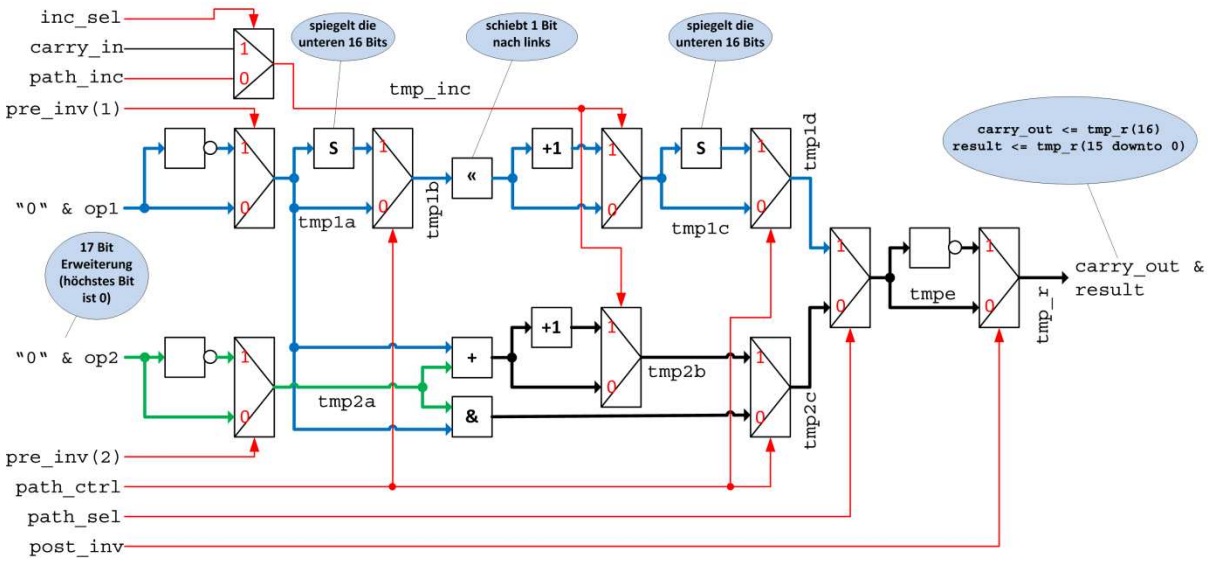

Abbildung 2: Strukturelle Darstellung der ALU.

In der Abbildung ist die ALU als Verknüpfung von einzelnen atomare Rechen- und Logik-Operationen durch Multiplexern (jeweils Teile des Schaltnetzes) gezeigt. Durch entsprechende Steuersignale (rot) wird der Datenfluss der Operanden-Signale (*op1* und *op2*) bzw. eines eingehenden Übertrags-bits (*carry_in*) durch die funktionalen Blöcke bis hin zu den Ausgangssignalen für die Ergebnisse Ergebnissignalen (*carry_out* und *result*) gesteuert. Intern arbeitet mit 17 Bit breiten Operanden/Signalen. Diese sind zwischen den Blöcken eingezeichnet (*1a*, *2a*, …).

Eine bestimmte Kombination der Steuersignale definiert die Betriebsart der ALU, wobei nicht alle Kombinationen sinnvoll sind. Tabelle 2 enthält symbolische Namen von einigen „sinnvollen" Betriebsarten, wie z.B. ADD, NOR, … .

Betriebsart `inst(11 … 8)`			**Decodierte Steuersignale (binär)**						
Bin.	**Hex.**	**Symbol**	`inc_sel`	`pre_inv(1)`	`pre_inv(2)`	`path_inc`	`path_ctrl`	`path_sel`	`post_inv`
"0000"	0	ADD	0	0	0	0		0	0
"0001"	1	AND	X	0	0	X	1	0	0
"0010"	2								
"0011"	3	ADC	1	0	0	X	1	0	0
"0100"	4	SUB	0	0	1	1	1	0	0
"0101"	5	NEG	0	1	0	1	1	0	0
"0110"	6	OR	X	1	1	X	0	0	1
"0111"	7	NOR	X	1	1	X	0	0	0
"1000"	8	SHR	0	0	X	0	1	1	0
"1001"	9	SHL	0	0	X	0	0	1	0
"1010"	A	RCR	1	0	X	X	1	1	0
"1011"	B	RCL	1	0	X	X	0	1	0
"1100"	C								
"1101"	D								
"1110"	E								
"1111"	F	BYPASS	X	X	X	X	X	X	X

Tabelle 2: Übersicht der Betriebsarten der ALU.

Erläuterungen:

- In der Betriebsart BYPASS wird die ALU nicht durch das Steuerwerk genutzt. Daher sind in dieser Betriebsart die decodierten Steuersignale beliebig.
- Leere Einträge bei den decodierten Steuersignale und der symbolischen Benennung zeigen nicht definierte Betriebsarten der ALU.

Steuerwerk

Das Steuerwerk arbeitet in 5 Phasen, wobei eine Phase einem Taktschritt entspricht. Somit benötigt jede Befehlsausführung 5 Taktschritte. Abbildung 3 beschreibt diesen Befehlszyklus.

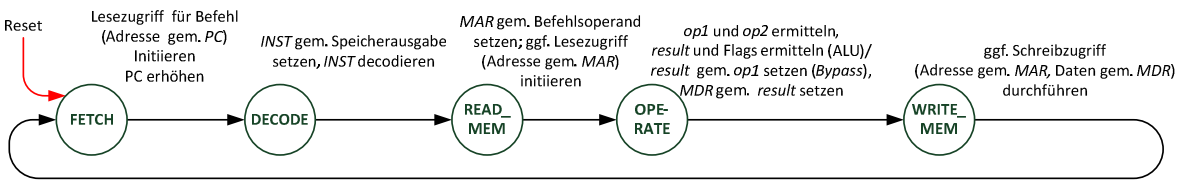

Abbildung 3: Befehlszyklus des Steuerwerks.

In Tabelle 3 sind die möglichen Betriebsarten des Steuerwerks zu finden. Die Betriebsarten unterscheiden sich Anhand der Operanden-Ursprungs, des Ziels für das Ergebnis und ggf. den Flags, von denen das Rückschreiben des Ergebnisses abhängt.

Betriebsart `inst(15 … 12)`			**Decodierte Steuersignale (symbolisch)**				
Binär	**Hexa-dezimal**	**Symbol**	**`src1`**	**`src2`**	**`dest`**	**`update_ condition`**	**`update_ carry`**
"0000"	**0**	**MODE1**	ACC	INST	ACC	ALWAYS	False
"0001"	**1**	**MODE2**	ACC	INST	ACC	ALWAYS	True
"0010"	**2**	**MODE3**	INST	NONE	ACC	ALWAYS	false
"0011"	**3**	**MODE4**	ACC	NONE	ACC	ALWAYS	false
"0100"	**4**	**MODE5**	INST	PC	PC	ALWAYS	false
"0101"	**5**	**MODE6**	INST	PC	PC	CARRY	false
"0110"	**6**	**MODE7**	INST	PC	PC	ZERO	false
"0111"	**7**						
"1000"	**8**	**MODE8**	AMEM	ACC	ACC	ALWAYS	false
"1001"	**9**	**MODE9**	ACC	AMEM	ACC	ALWAYS	true
"1010"	**A**	**MODE10**	ACC	NONE	AMEM	ALWAYS	false
"1011"	**B**	**MODE11**	RMEM	ACC	ACC	ALWAYS	false
"1100"	**C**	**MODE12**	ACC	RMEM	ACC	ALWAYS	true
"1101"	**D**	**MODE13**	ACC	NONE	RMEM	ALWAYS	false
"1110"	**E**	**MODE14**	SMEM	NONE	IREG	ALWAYS	false
"1111"	**F**	**MODE15**	IREG	NONE	SMEM	ALWAYS	false

Tabelle 3: Übersicht der Betriebsarten des Steuerwerks.

Erläuterungen:

- *src1*,*src2* und *dest* geben die Quellen der Operanden *op1* und *op2* (vgl. ALU) bzw. das Ziel des Ergebnisses an
  - *ACC*: Akkumulator-Register
  - *PC*: Befehlszeiger
  - *INST*: Bildet einen Direktoperand aus dem Befehlswort, wobei die unteren 8 Bit auf den Wert des Befehlsoperanden und die oberen 0 bit auf 0 gesetzt werden.
  - *AMEM*: Absolute Speicheradresse (gem. Operanden-Feld des Befehlswortes)
  - *RMEM*: PC-relative Speicheradresse (gem. Operanden-Feld des Befehlswortes) im Zweierkomplement (PC+RMEM+1=AMEM)
  - *SMEM*: Stack-Speicher (Adresse gem. SP)
  - *IREG*: Indiziertes Register (0=PC, 1=ACC)
  - *NONE*: Operand wird gleich 0 gesetzt
- *update_cond* legt fest, ob das Ergebnis immer (ALWAYS) oder nur bei entsprechend gesetztem Flag in das Ergebnisregister/das RAM zurückgeschrieben werden soll.
- *update_carry* legt fest, ob das Carry-Flag gem. der Ausgabe der ALU aktualisiert werden soll.**:**
- Das Zero-Flag wird bei jedem Befehl aktualisiert. Ist der Akkumulator gleich Null, dann wird es gesetzt und andernfalls zurückgesetzt.