Übersicht

Übersicht

1. Allgemeine Grundlagen

2. Spezielle Grundlagen

3. Vorversuche

4. Versuche

Übersicht

1. Allgemeine Grundlagen

2. Spezielle Grundlagen

3. Vorversuche

4. Versuche

Übersicht

1. Allgemeine Grundlagen

2. Spezielle Grundlagen

3. Vorversuche

4. Versuche

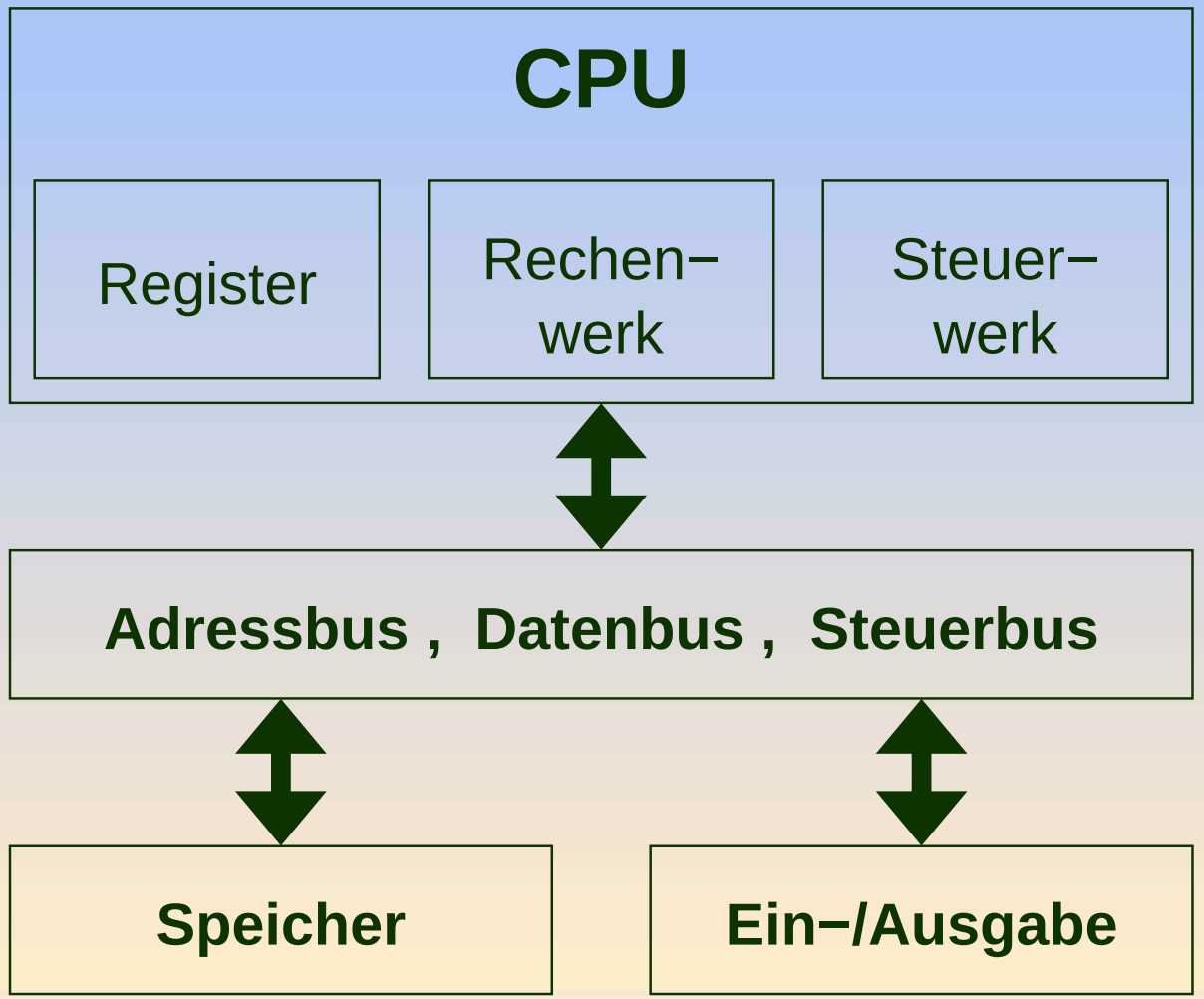
Grundlagen: Von-Neumann-Architektur

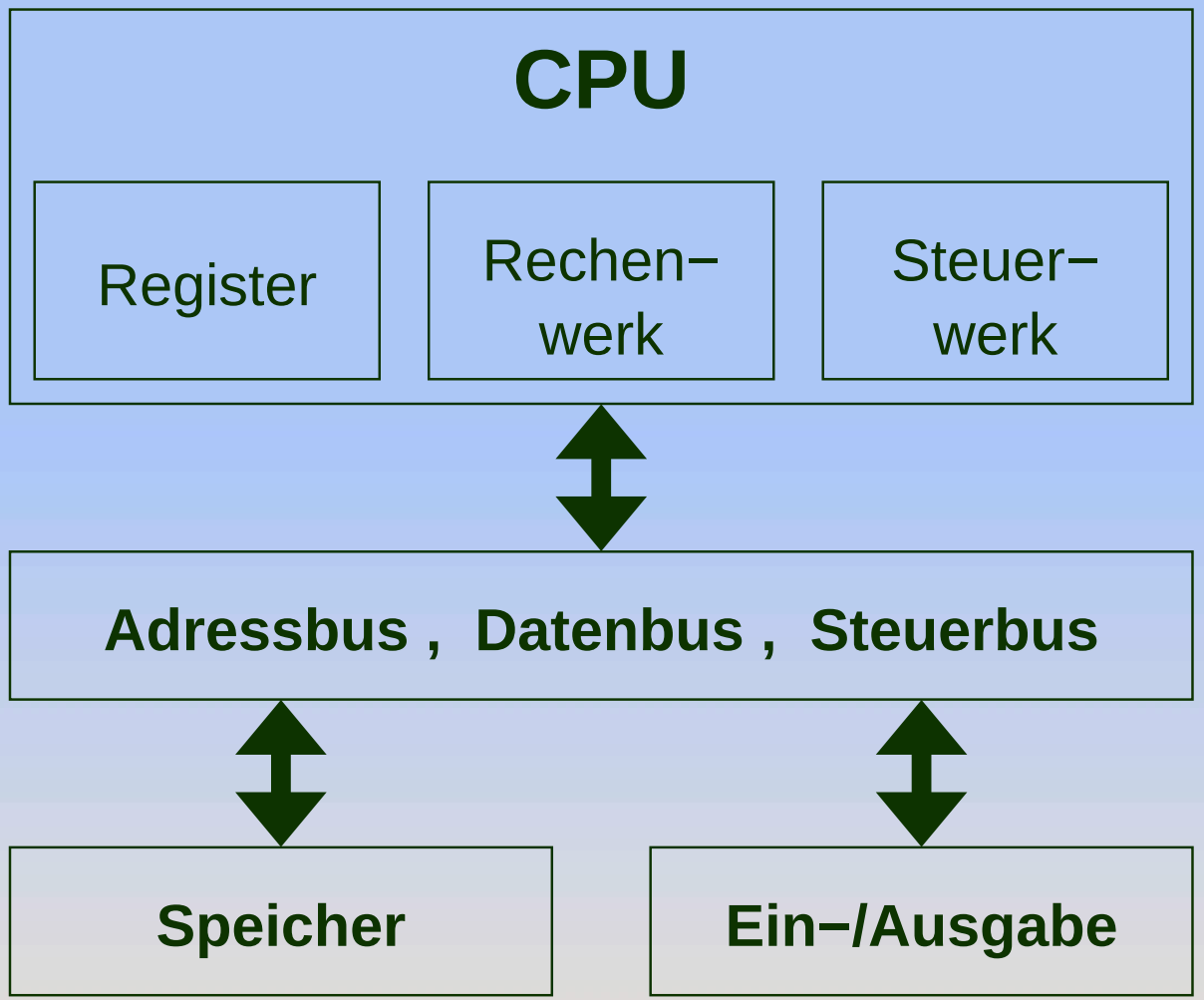
Grundlagen: Von-Neumann-Architektur

– John von Neumann (geboren als Johann von Neumann), 1903-1957

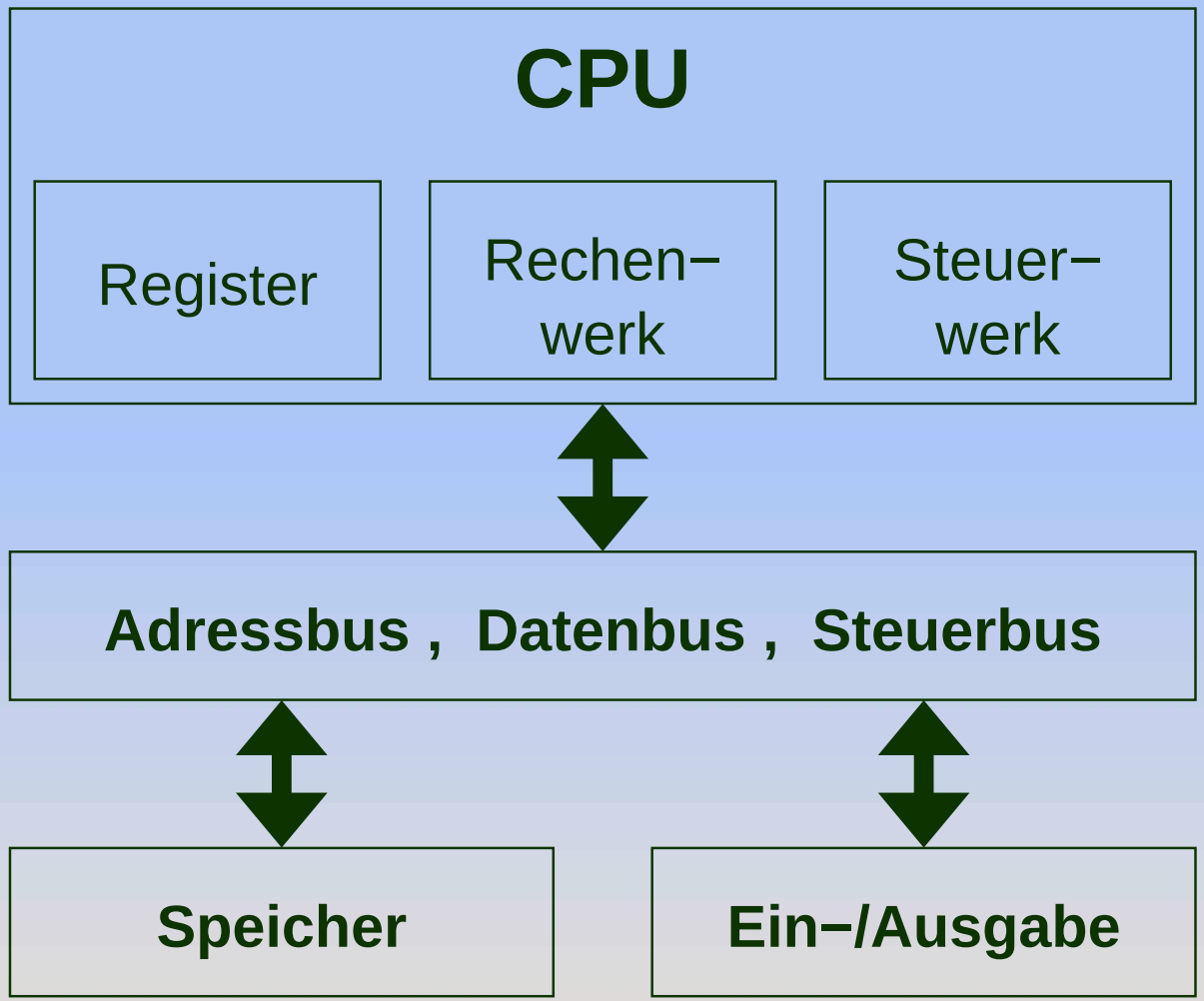
Grundlagen: Von-Neumann-Architektur

– John von Neumann (geboren als Johann von Neumann), 1903-1957



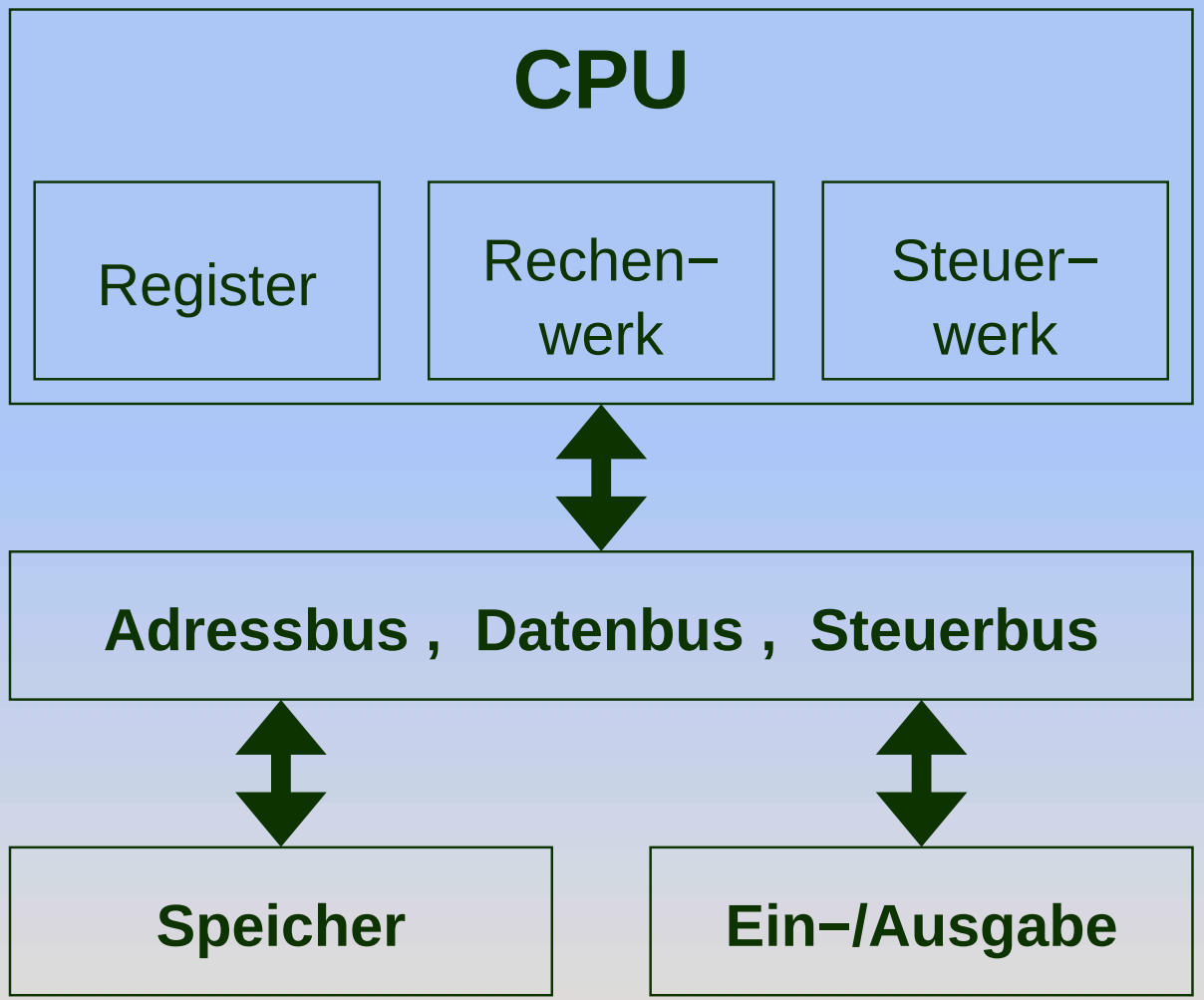


– Baugruppen: CPU, Busse, Speicher, Ein-/Ausgabebausteine



– Baugruppen: CPU, Busse, Speicher, Ein-/Ausgabebausteine

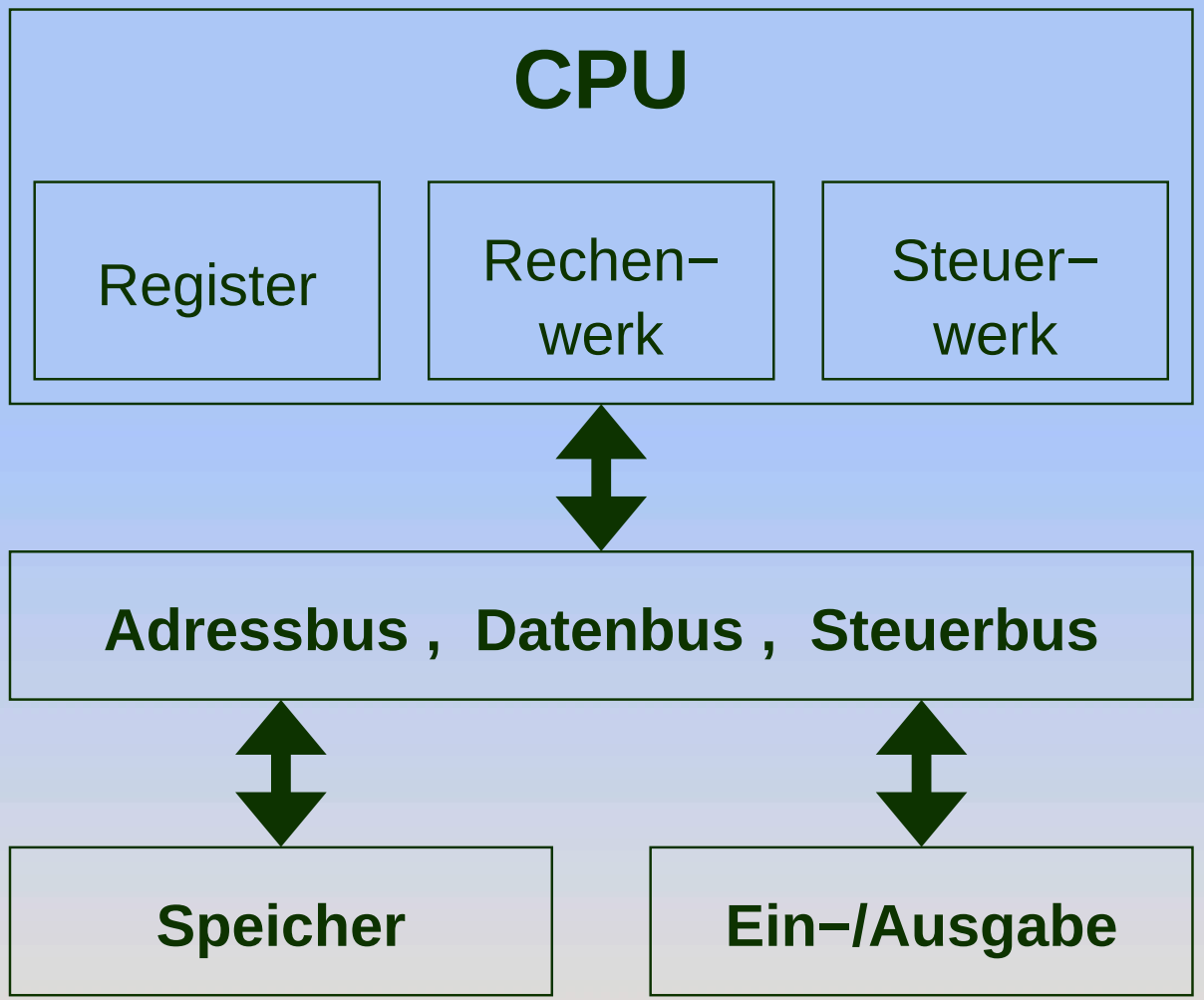
– Zahlendarstellung in der Regel binär



– Baugruppen: CPU, Busse, Speicher, Ein-/Ausgabebausteine

– Zahlendarstellung in der Regel binär

– Universalität (Architektur unabhängig von Anwendung, Programmierbarkeit)



– Baugruppen: CPU, Busse, Speicher, Ein-/Ausgabebausteine
– Zahlendarstellung in der Regel binär
– Universalität (Architektur unabhängig von Anwendung, Programmierbarkeit)
– Speicher (Daten und Programm, Adressierung)

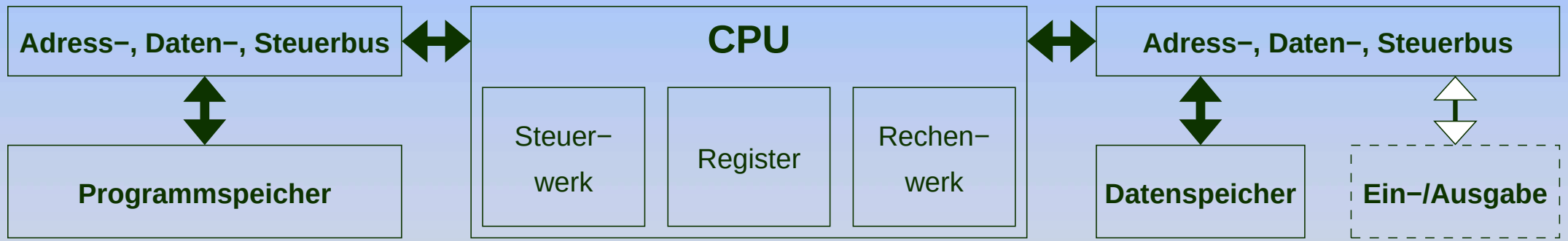
Grundlagen: Harvard-Architektur

Grundlagen: Harvard-Architektur

– von Howard Aiken an der Universität Harvard konstruierte “Mark I” (1937, 1939-44)

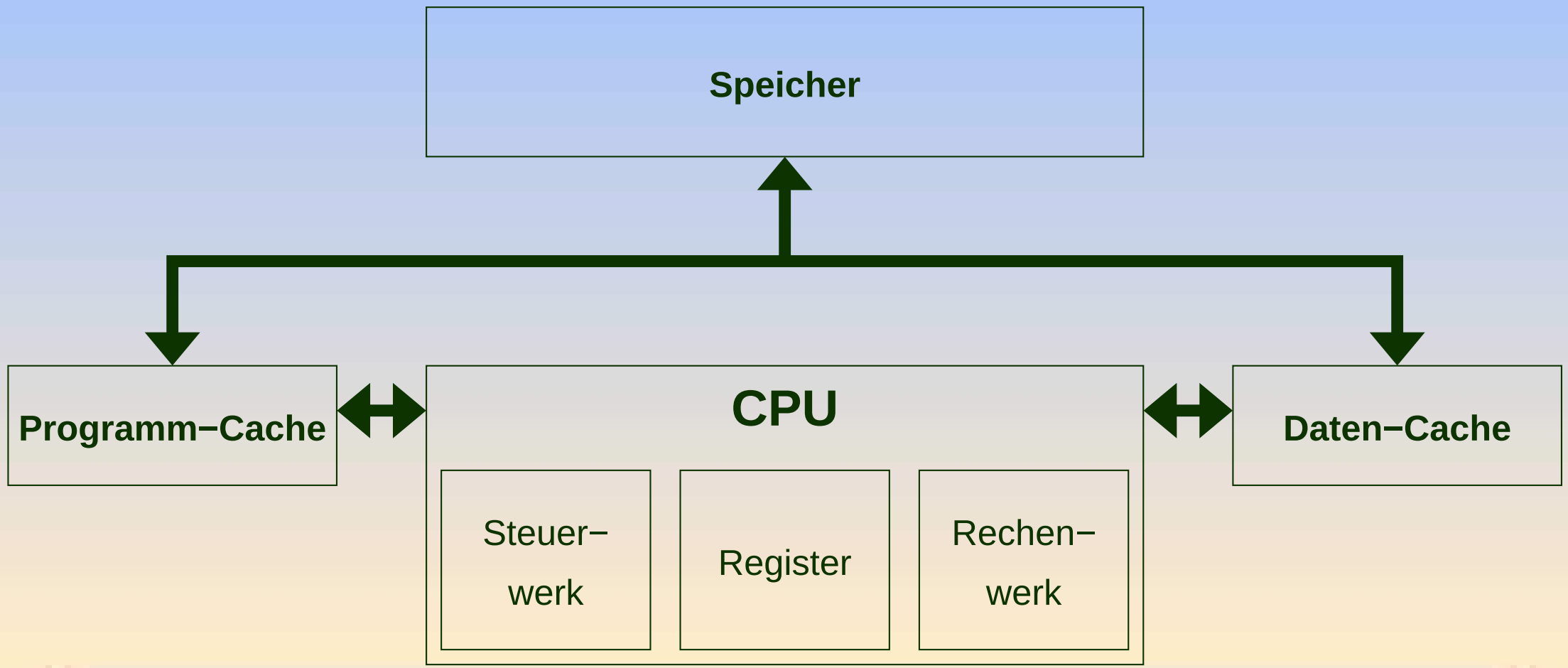
Grundlagen: Harvard-Architektur

– von Howard Aiken an der Universität Harvard konstruierte “Mark I” (1937, 1939-44)



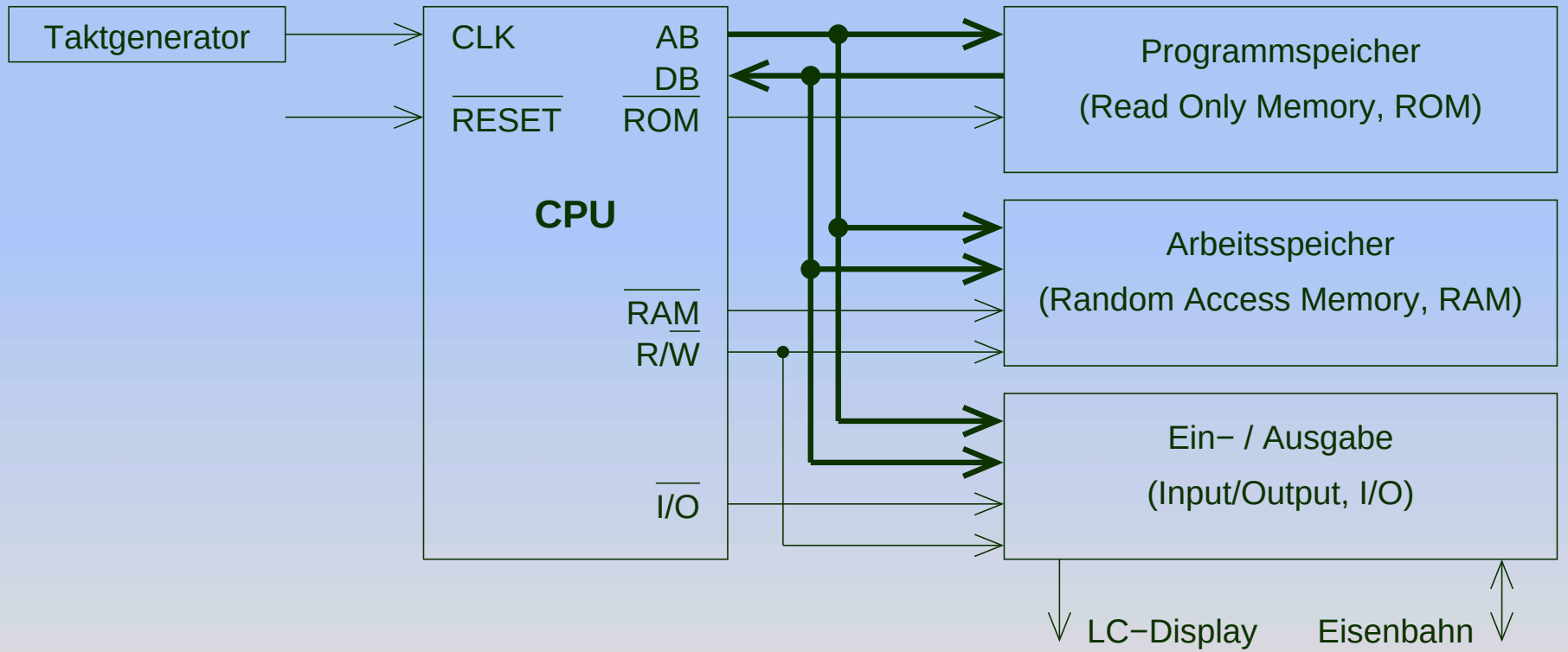
Grundlagen: Harvard-Architektur

– von Howard Aiken an der Universität Harvard konstruierte “Mark I” (1937, 1939-44)

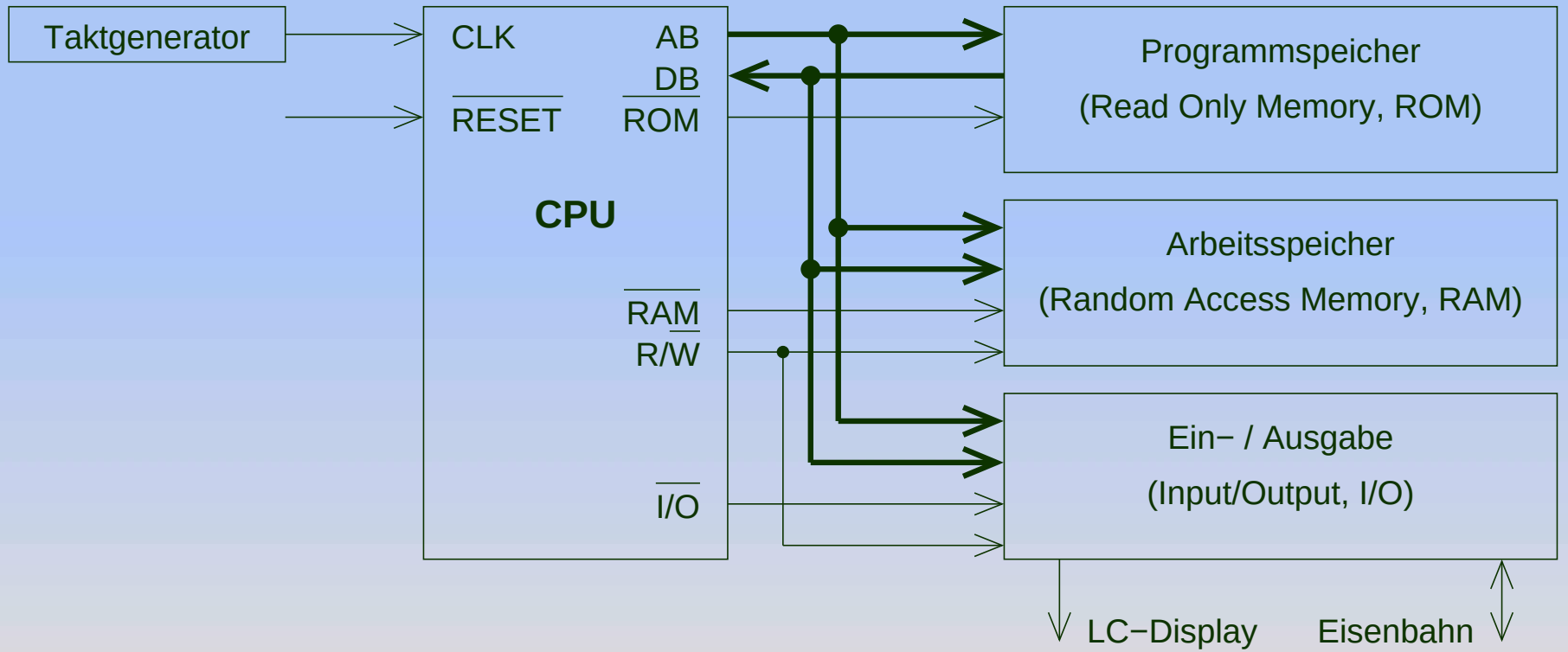


DEP: Architektur Gesamtsystem

DEP: Architektur Gesamtsystem

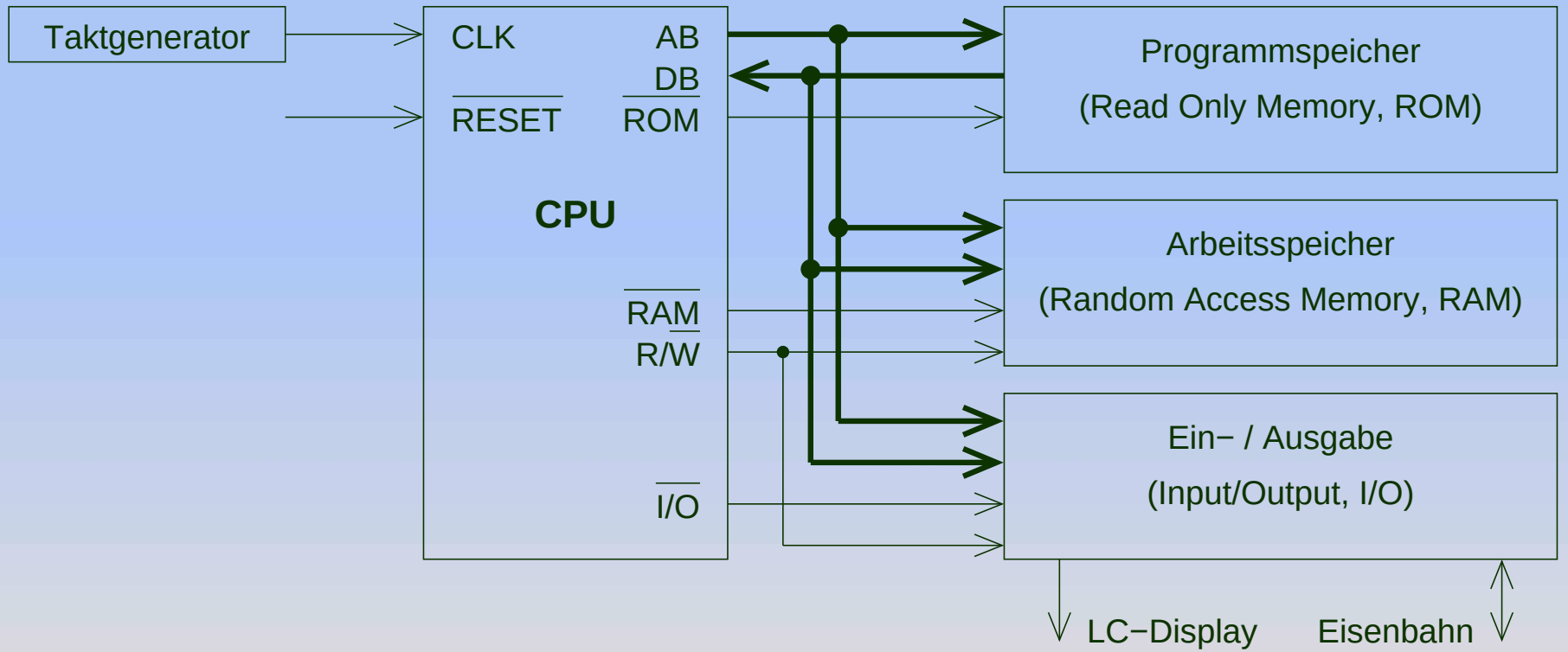


DEP: Architektur Gesamtsystem



Verarbeitung von Maschinenbefehlen:

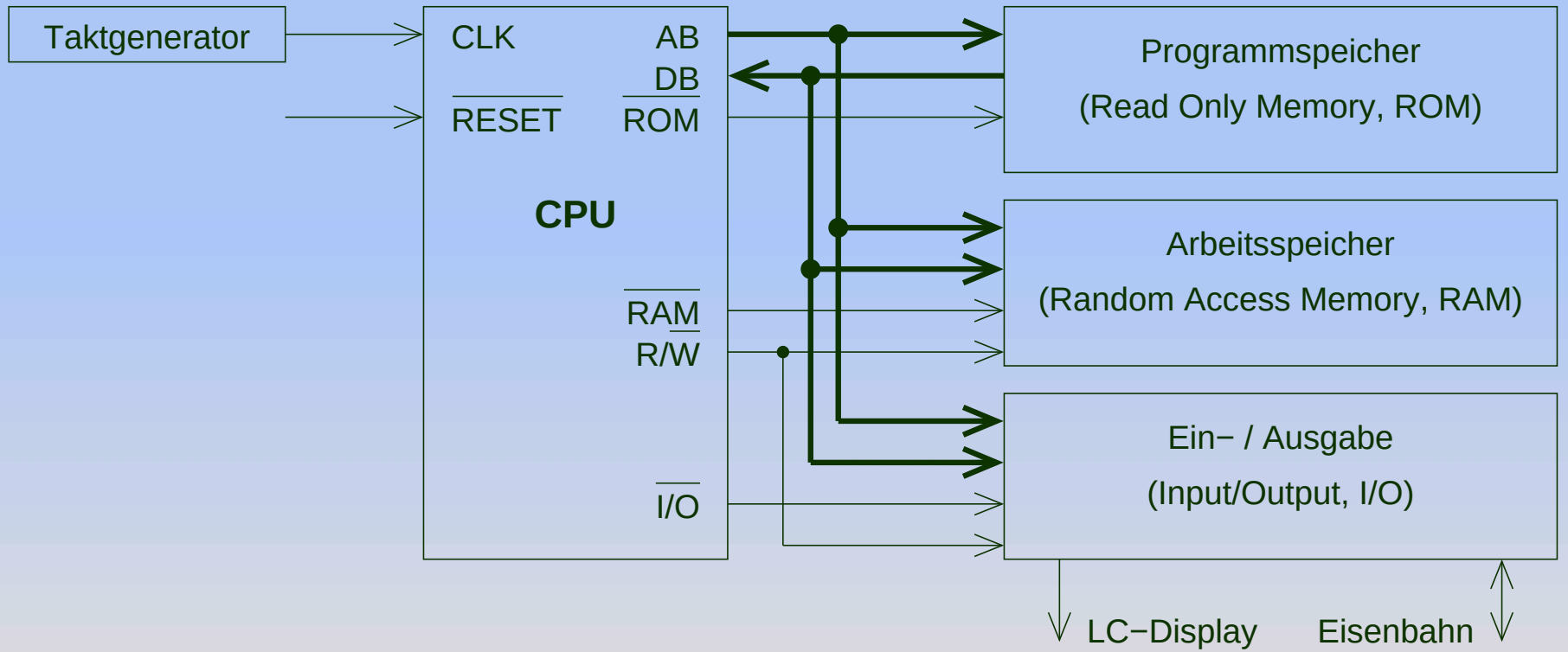
DEP: Architektur Gesamtsystem



Verarbeitung von Maschinenbefehlen:

– Laden/Speichern RAM

DEP: Architektur Gesamtsystem

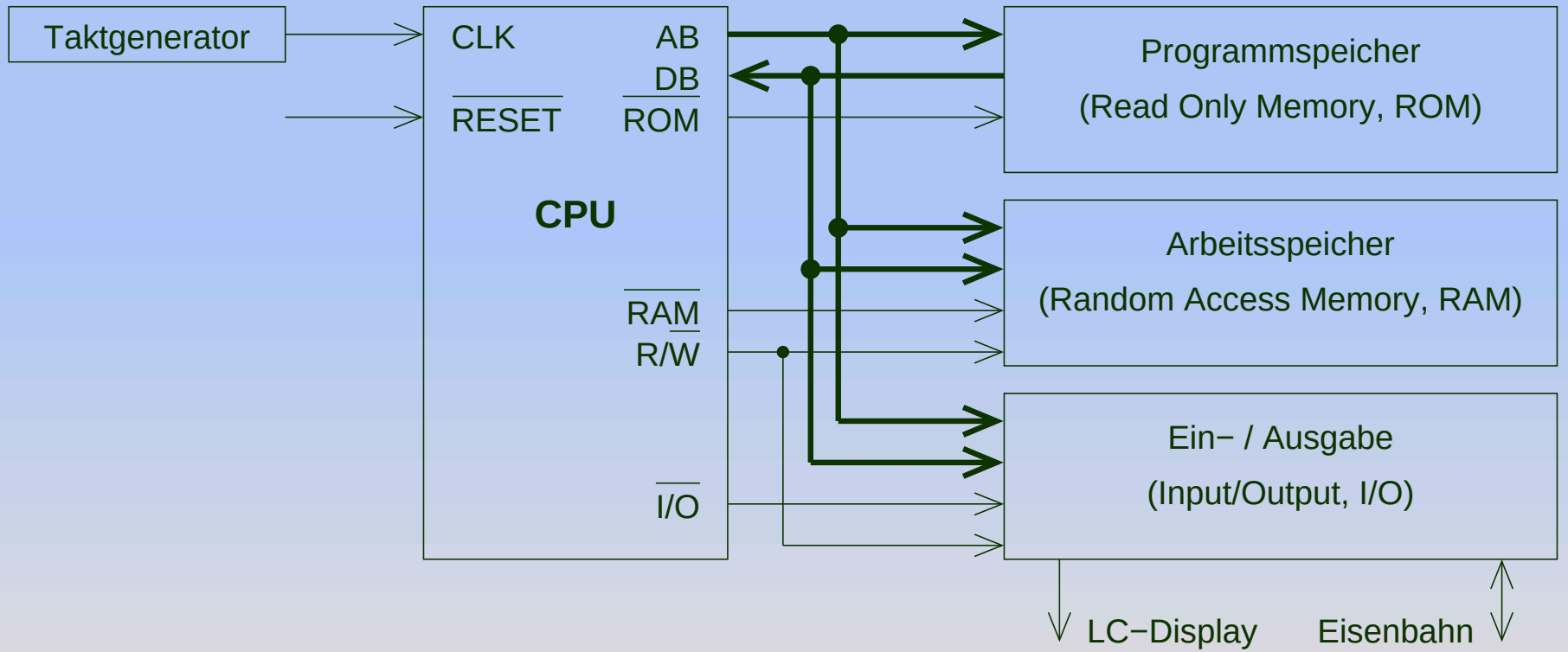


Verarbeitung von Maschinenbefehlen:

– Laden/Speichern RAM

– Ein-/Ausgabe

DEP: Architektur Gesamtsystem



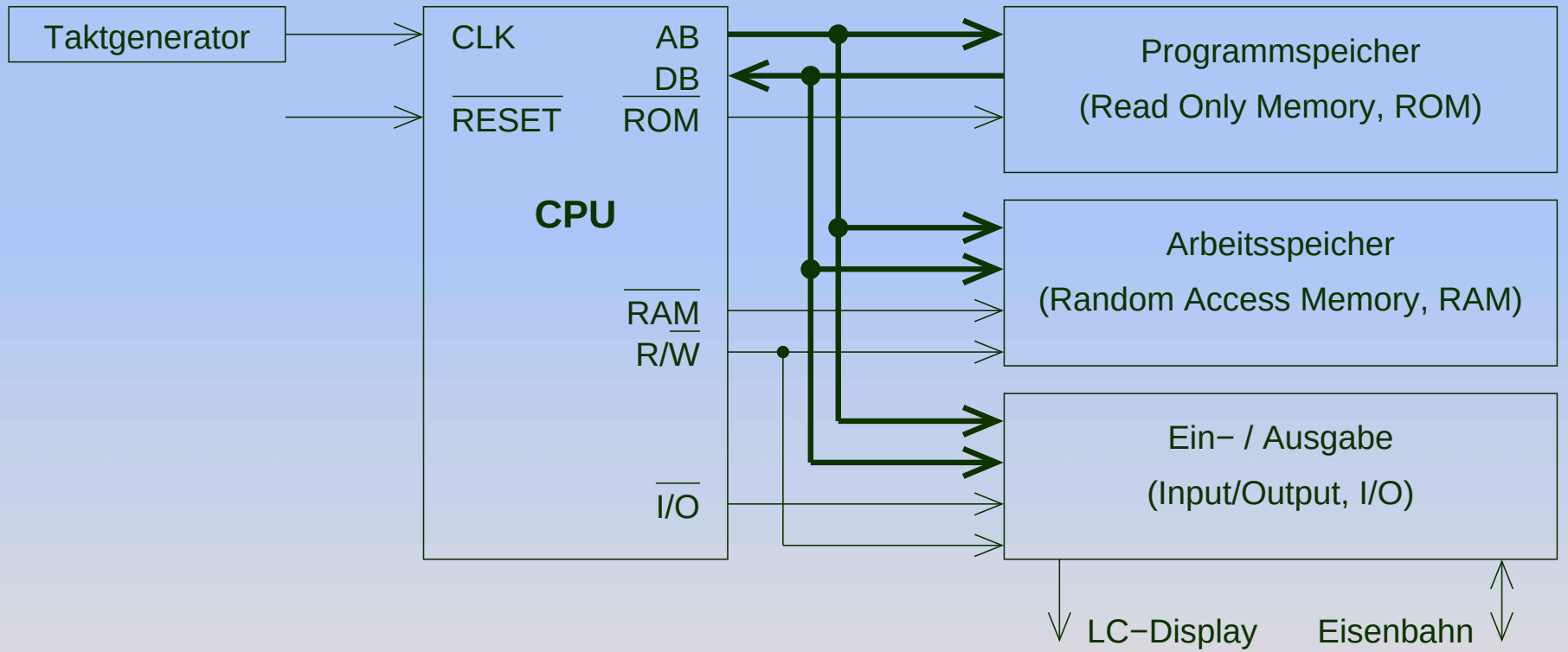
Verarbeitung von Maschinenbefehlen:

– Laden/Speichern RAM

– Ein-/Ausgabe

– Rechenoperationen

DEP: Architektur Gesamtsystem



Verarbeitung von Maschinenbefehlen:

– Laden/Speichern RAM

– Ein-/Ausgabe

– Rechenoperationen

– Bedingte/unbedingte Programmverzweigungen

DEP: Instruktionssatz

DEP: Instruktionssatz

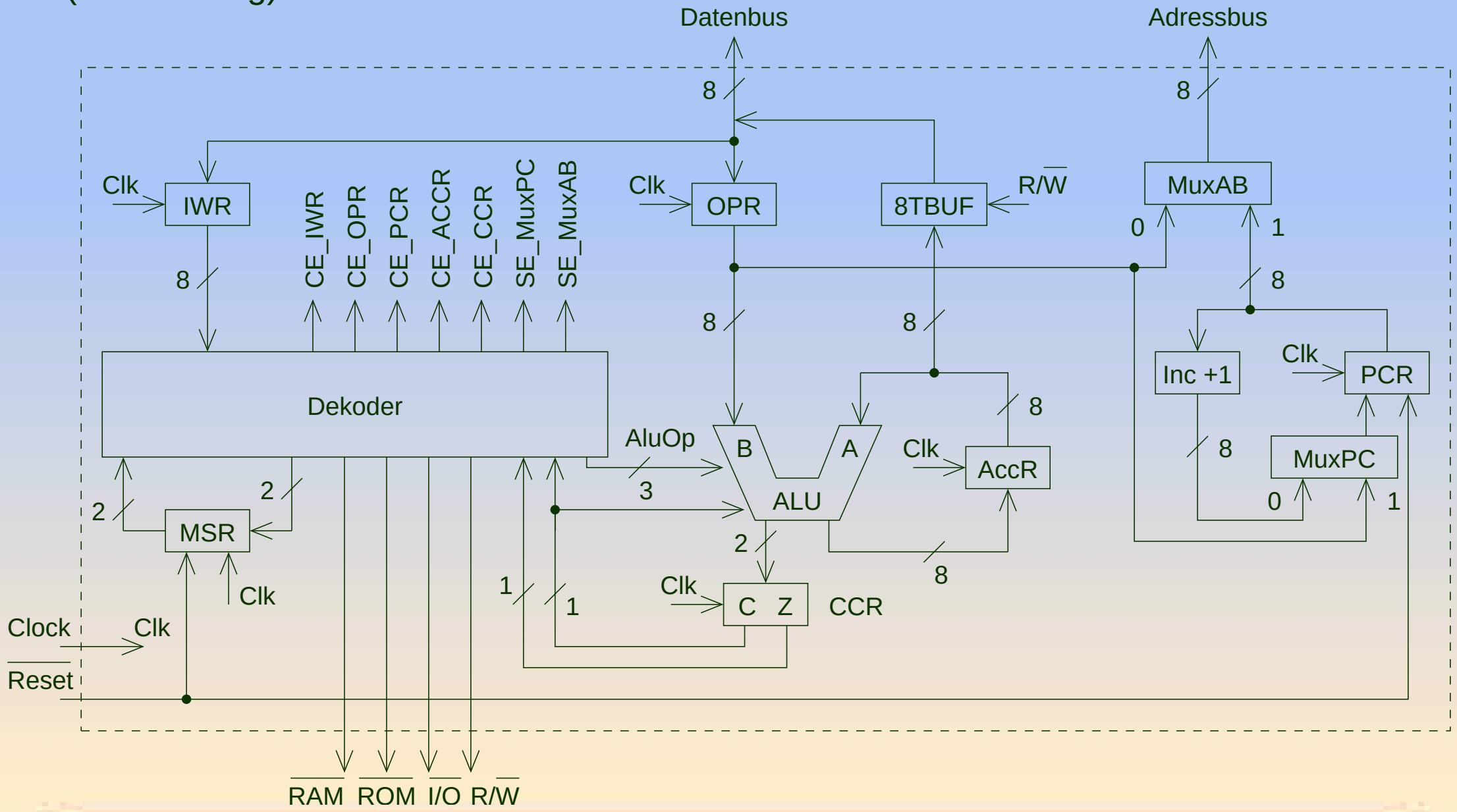
Sedezimalcode	Befehlskürzel	Funktion	beeinflusste Flags
Speichertransferbefehle			
80 xx	STA xx	Speichere Inhalt von A an Adresse xx	
A0 zz	STAX zz	Speichere Inhalt von A an Adresse xx, die an Adresse zz steht	
40 xx	LDA xx	Lade A mit Inhalt an Adresse xx	Z
48 yy	LDAI yy	Lade A mit Konstante yy	Z
Sprungbefehle			
28 dd	BCS dd	Wenn Carry-Flag gesetzt: PCR:=dd	
18 dd	BEQ dd	Wenn Zero-Flag gesetzt: PCR:=dd	
30 xx	JMP xx	Setze PCR auf den Inhalt an Adresse xx	
38 dd	JMPI dd	Setze PCR auf dd	
Ein-/Ausgabe-Befehle			
50 xx	IN xx	Lade A mit Inhalt an Portadresse xx	Z
90 xx	OUT xx	Schreibe Inhalt von A an Portadresse xx	
Rechenwerksbefehle (Speicher)			
41 xx	ADD xx	Addiere A zu Inhalt von xx	C und Z
42 xx	OR xx	Bitweises Oder von A und [xx]	Z
43 xx	SBC xx	A := A - [xx] - C	C und Z
44 xx	XOR xx	Bitweises ExOder von A und [xx]	Z
45 xx	ADC xx	A := A + [xx] + C	C und Z
46 xx	AND xx	Bitweises Und von A und [xx]	Z
47 xx	SETF xx	Setze C und Z entsprechend Bit 6 und 7 von [xx] und behalte A unverändert bei	C und Z
Rechenwerksbefehle (Immediate, also konstanter Operand)			
49 yy	ADDI yy	Addiere A zu yy	C und Z
4A yy	ORI yy	Bitweises Oder von A und yy	Z
4B yy	SBCI yy	A := A - yy - C	C und Z
4C yy	XORI yy	Bitweises ExOder von A und yy	Z
4D yy	ADCI yy	A := A + yy + C	C und Z
4E yy	ANDI yy	Bitweises Und von A und yy	Z
4F yy	SETFI yy	Setze C und Z entsprechend Bit 6 und 7 von yy und behalte A unverändert bei	C und Z

DEP: CPU-Architektur*

(*: Vorschlag)

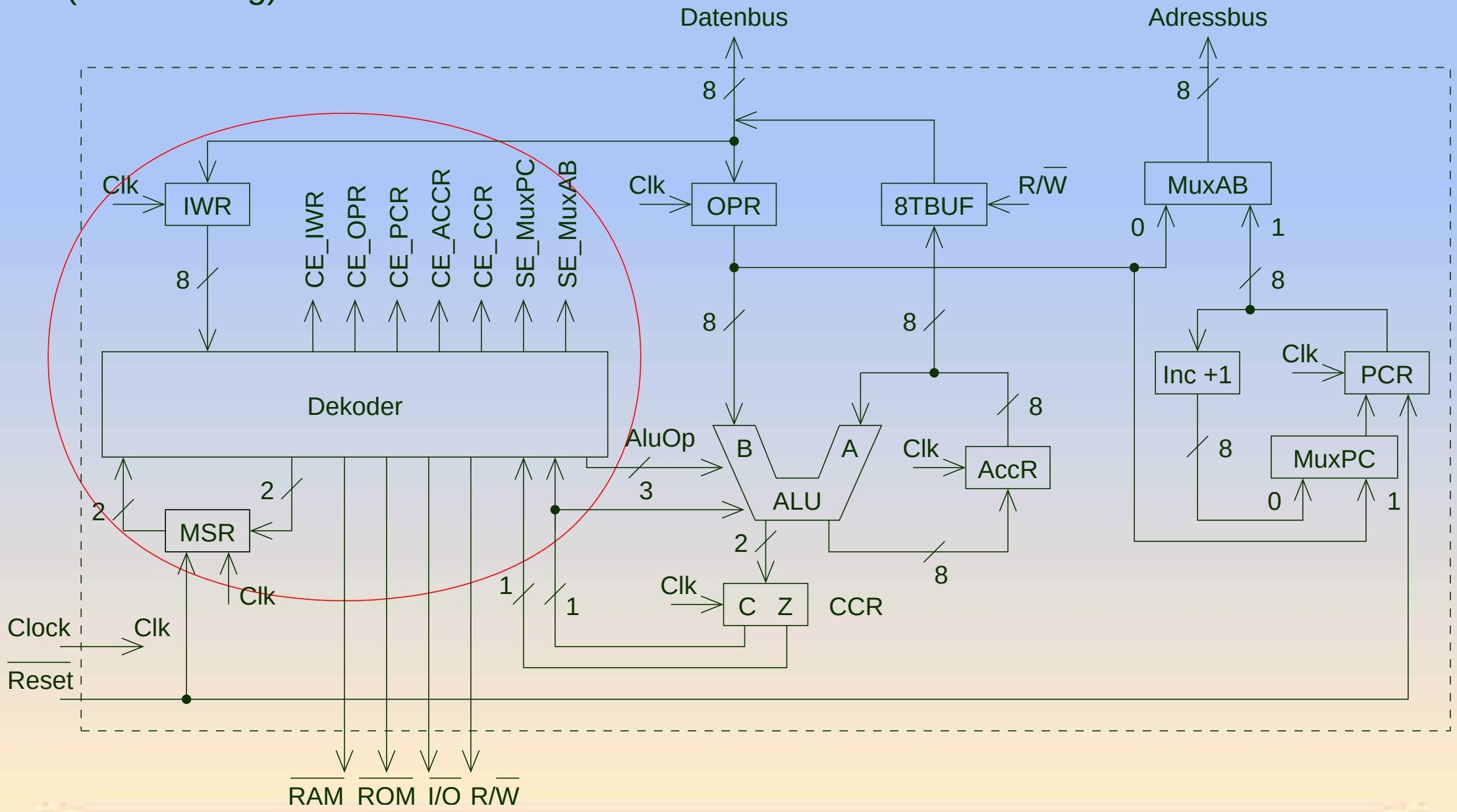
DEP: CPU-Architektur*

(*: Vorschlag)



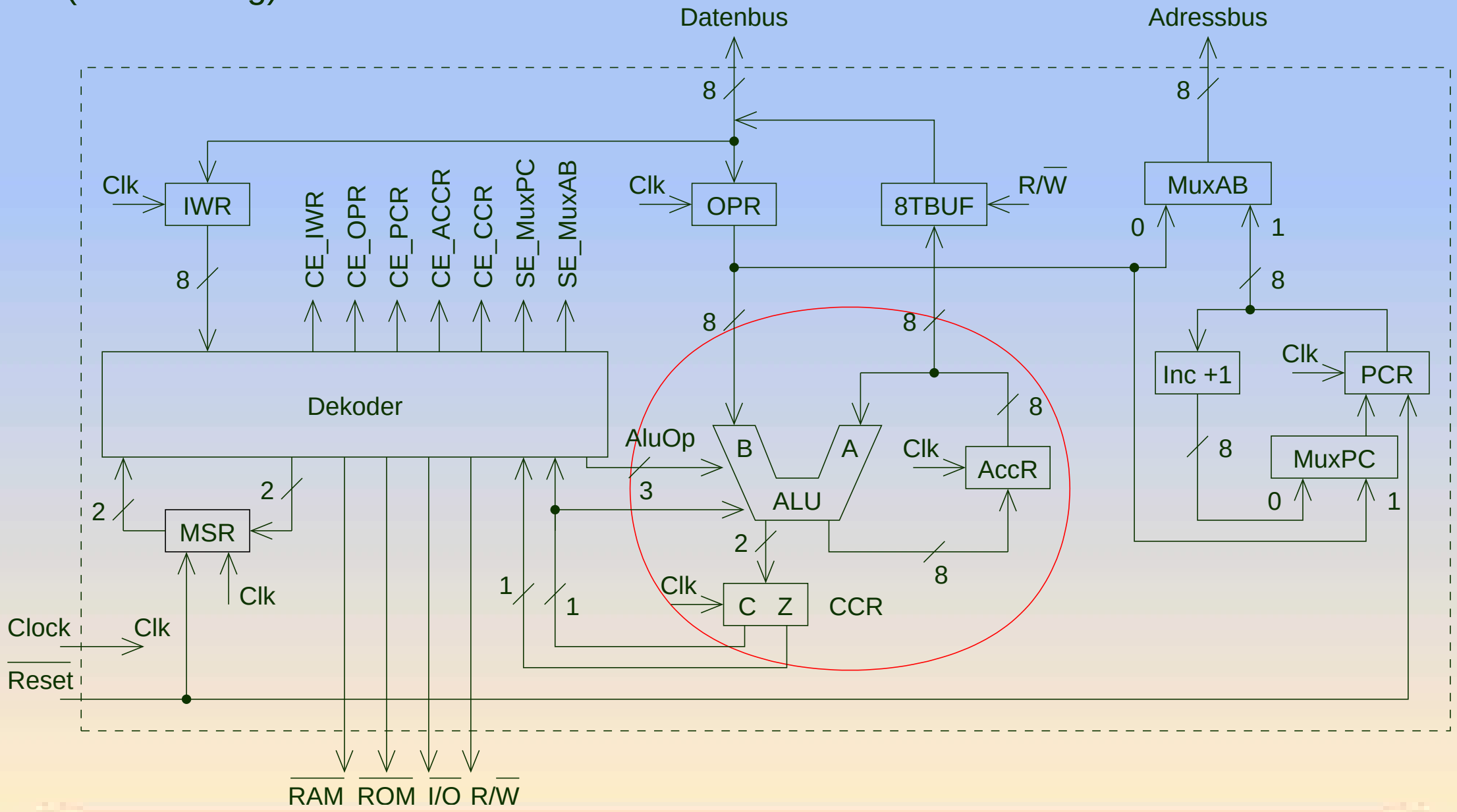
DEP: CPU-Architektur*

(*: Vorschlag)



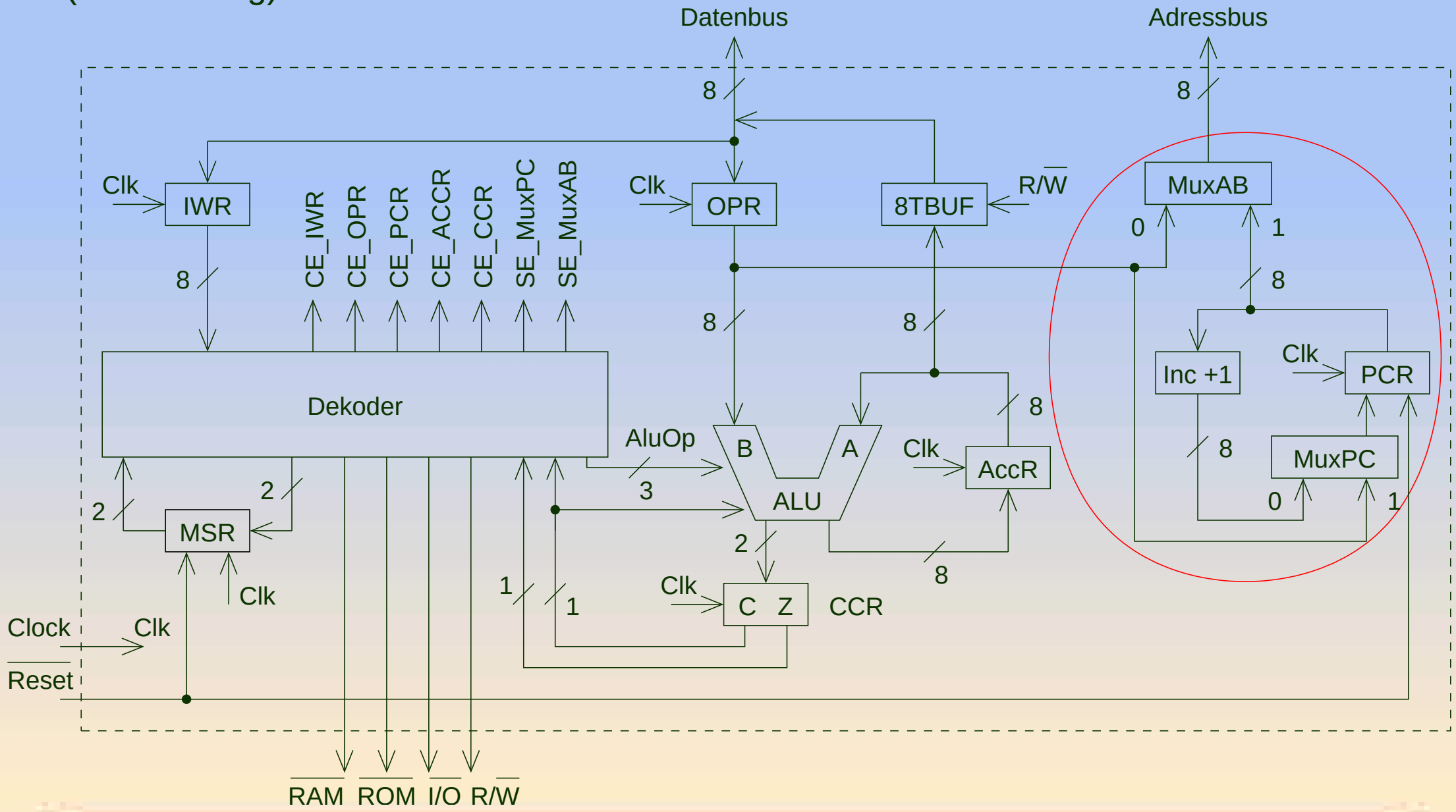
DEP: CPU-Architektur*

(*: Vorschlag)



DEP: CPU-Architektur*

(*: Vorschlag)



Dekodertabelle

Dekodertabelle

Inst.	Instruction Word								CCR		MSR	MSR	IWR	OPR	Mux	Mux	AccR	CCR	PCR	$\overline{\text{RAM}}$	$\overline{\text{ROM}}$	$\overline{\text{IO}}$	R/$\overline{\text{W}}$	Alu-
	ST	LD	BCS	BEQ	IMM	AI2	AI1	AI0	Z	C	out	in			AB	PC								Op
	–	–	–	–	–	–	–	–	–	–	0	1	1	0	1	0	0	0	1	1	0	1	1	–
	–	–	–	–	–	–	–	–	–	–	1	2	0	1	1	0	0	0	1	1	0	1	1	–
STA	1	0	0	0	0	0	0	0	–	–	2	0	0	0	0	–	0	0	0	0	1	1	0	–
STAX	1	0	1	0	0	0	0	0	–	–	2													
	1	0	1	0	0	0	0	0	–	–	3													
LDA	0	1	0	0	0	0	0	0	–	–	2													
	0	1	0	0	0	0	0	0	–	–	3													
LDAI	0	1	0	0	1	0	0	0	–	–	2													
BCS	0	0	1	0	1	0	0	0	–	0	2													
	0	0	1	0	1	0	0	0	–	1	2													
BEQ	0	0	0	1	1	0	0	0	0	–	2													
	0	0	0	1	1	0	0	0	1	–	2													
JMP	0	0	1	1	0	0	0	0	–	–	2													
	0	0	1	1	0	0	0	0	–	–	3													
JMPI	0	0	1	1	1	0	0	0	–	–	2													
IN	0	1	0	1	0	0	0	0	–	–	2													
	0	1	0	1	0	0	0	0	–	–	3													
OUT	1	0	0	1	0	0	0	0	–	–	2													
Alu	0	1	0	0	0	a	b	c	–	–	2													
	0	1	0	0	0	a	b	c	–	–	3													
AluI	0	1	0	0	1	a	b	c	–	–	2													

Vorversuche

Vorversuche

Vorbereitungsaufgabe: Vorbereitung der DEP-CPU-Entwicklung

1. Machen Sie sich durch Studium des Theorieteils mit der Architektur und dem Befehlssatz der DEP-CPU vertraut.
2. Ergänzen Sie die Decodertabelle.
3. Entwerfen Sie zur Übung den Inkrementer “Inc + 1” (falls noch nicht in Versuch 5 geschehen) mit möglichst geringem Ressourcen-Verbrauch. Sie können die von Ihnen bereits entwickelten Blöcke “Halbaddierer” und “Volladdierer” wiederverwenden. Erstellen Sie den Entwurf zunächst auf Papier.

Versuche

Versuche

Praktikumsaufgabe: Entwicklung und Test der DEP-CPU

1. Machen Sie sich durch Studium der Unterlagen (im Praktikumsraum vorhanden) mit dem FPGA-Board und der Software “Quartus II” vertraut.
2. Entwerfen und testen Sie den gesamten Prozessor. Beginnen Sie zur Übung zweckmäßig mit dem Inkrementer “Inc + 1”, erstellen Sie danach die weiteren kleinen Baugruppen (Multiplexer, Register) - sofern dies noch nicht in Versuch 5 geschehen ist. Dann können Sie sich die ALU vornehmen und schließlich den Decoder. Sie erhalten als Basis wieder eine Projektdatei von Ihrem Tutor.
   Erstellen Sie Ihre CPU dabei nur innerhalb der “CPU”-Schematic-Datei bzw. in darin eingefügten Blöcken (= Dateien).
3. Zum Test können Sie wieder den Simulator in der Entwicklungsumgebung “Quartus II” verwenden. Sollte Ihr Design im Simulator fehlerfrei sein, nicht jedoch auf dem FPGA-Board laufen, so achten Sie auf die Einstellung “Timing”-Simulation im “Simulator-Tool” (siehe Quartus-Kurzbeschreibung). Diese Simulation ist zuverlässiger als die einfache “Functional”-Simulation.
4. Um die Funktionsfähigkeit Ihres Prozessors nachzuweisen, wird Ihnen von Ihrem Tutor ein Beispielprogramm vorgegeben, welches fehlerfrei abgearbeitet werden muss.

5. Schreiben Sie ein Assembler-Programm, das – ähnlich der Modelleisenbahn-Automaten-Aufgabe aus dem letzten Versuch – einen Zug mit *drei* Waggons zum Entkuppler zieht und dann nacheinander die Waggons abkuppelt und auf die Gleise 1 bis 3 verteilt.
   Sie können über einen IN-Befehl an Adresse 80 (hexadezimal) die Reed-Kontakte abfragen und über einen OUT-Befehl (gleiche Adresse) Fahrstrom, Fahrtrichtung, den Entkuppler und die Weiche steuern. Lesen Sie hierzu nochmals die Beschreibung zum letzten Versuch.
   Die Bits sind folgendermaßen angeordnet:

D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
res.	res.	R345	R5	R4	R3	R2	R1

Lesen

D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
res.	res.	res.	W2	W1	ENTK	RICH	FS

Schreiben

   (res. = reserviert: Sowohl “High” als auch “Low” möglich)

6. Fragen Sie zum Erstellen und Übersetzen des Assembler-Programms Ihren Tutor. Er wird das Programm auch prüfen und mit Ihnen zusammen die Kompilierung Ihrer CPU und Konfiguration des FPGA-Boards vornehmen.

7. Bauen Sie die Eisenbahn in Absprache mit dem Tutor auf und testen Sie Ihr Programm.

8. Welche Modifikationen sind beim Entwurf einer 16-Bit CPU im Blockschaltbild der 8-Bit CPU bei Beibehaltung des 8-Bit Datenbusses und des Instruktionssatzes notwendigerweise vorzunehmen?
   Geben Sie sinnvolle Ergänzungen des Instruktionssatzes und die dafür benötigten Erweiterungen der zuvor modifizierten CPU-Stuktur an.

Inst.	Instruction Word								CCR		MSR	MSR	IWR	OPR	Mux	Mux	AccR	CCR	PCR	¬RAM	¬ROM	¬IO	R/¬W	Alu-
	ST	LD	BCS	BEQ	IMM	AI2	AI1	AI0	Z	C	out	in			AB	PC								Op
	–	–	–	–	–	–	–	–	–	–	0	1	1	0	1	0	0	0	1	1	0	1	1	–
	–	–	–	–	–	–	–	–	–	–	1	2	0	1	1	0	0	0	1	1	0	1	1	–
STA	1	0	0	0	0	0	0	0	–	–	2	0	0	0	0	–	0	0	0	0	1	1	0	–
STAX	1	0	1	0	0	0	0	0	–	–	2													
	1	0	1	0	0	0	0	0	–	–	3													
LDA	0	1	0	0	0	0	0	0	–	–	2													
	0	1	0	0	0	0	0	0	–	–	3													
LDAI	0	1	0	0	1	0	0	0	–	–	2													
BCS	0	0	1	0	1	0	0	0	–	0	2													
	0	0	1	0	1	0	0	0	–	1	2													
BEQ	0	0	0	1	1	0	0	0	0	–	2													
	0	0	0	1	1	0	0	0	1	–	2													
JMP	0	0	1	1	0	0	0	0	–	–	2													
	0	0	1	1	0	0	0	0	–	–	3													
JMPI	0	0	1	1	1	0	0	0	–	–	2													
IN	0	1	0	1	0	0	0	0	–	–	2													
	0	1	0	1	0	0	0	0	–	–	3													
OUT	1	0	0	1	0	0	0	0	–	–	2													
Alu	0	1	0	0	0	a	b	c	–	–	2													
	0	1	0	0	0	a	b	c	–	–	3													
AluI	0	1	0	0	1	a	b	c	–	–	2													