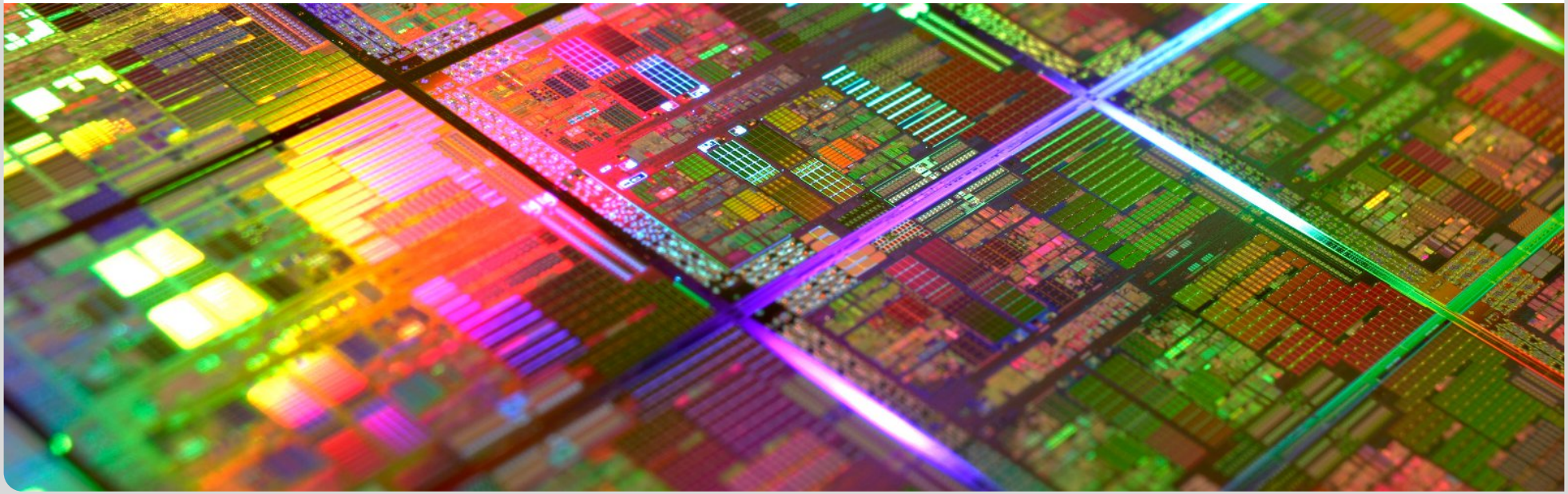
Karlsruher Institut für Technologie

Tutorium Rechnerorganisation

Woche 4

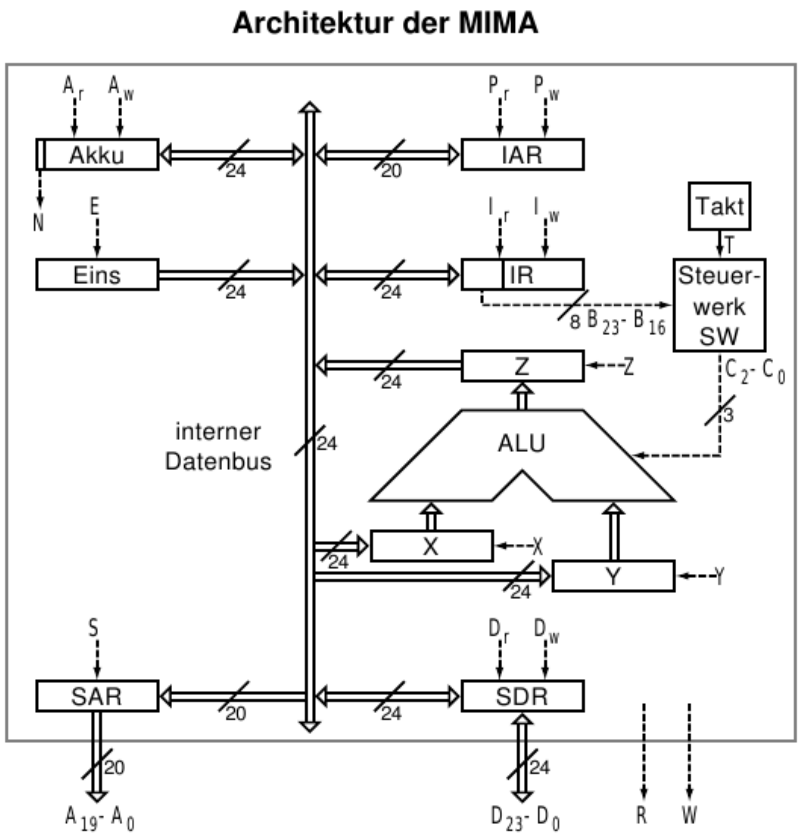
Tutorien 3 und 4 zur Vorlesung Rechnerorganisation



Heute

- Wiederholung der MIMA-Architektur
- MIMA-Befehl JIND (Jump Indirect)
- MIMA-Befehl JMS (Jump Subroutine)
- Wochentagsberechnung in MIMA

Die Architektur der MIMA



Register

Akku: Akkumulator
X: 1. ALU Operand
Y: 2. ALU Operand
Z: ALU Ergebnis
Eins: Konstante 1
IAR: Instruktionsadreßregister
IR: Instruktionsregister
SAR: Speicheradreßregister
SDR: Speicherdatenregister

Steuersignale vom SW

– für den internen Datenbus

A_r: Akku liest
A_w: Akku schreibt
X: X-Register liest
Y: Y-Register liest
Z: Z-Register schreibt
E: Eins-Register schreibt
P_r: IAR liest
P_w: IAR schreibt
I_r: IR liest
I_w: IR schreibt
D_r: SDR liest
D_w: SDR schreibt
S: SAR liest

– für die ALU

C_2- C_0: Operation auswählen

– für den Speicher

R: Leseanforderung
W: Schreibanforderung

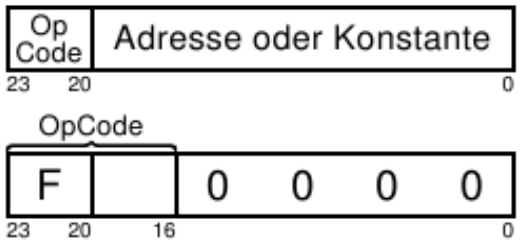
Meldesignale zum SW

T: Takteingang
N: Vorzeichen des Akku
B_{23}- B_{16}: OpCode-Feld im IR

Befehle der MIMA

OpCode	*Mnemonik*	*Beschreibung*
0	LDC c	c -> Akku
1	LDV a	<a> -> Akku
2	STV a	Akku -> <a>
3	ADD a	Akku + <a> -> Akku
4	AND a	Akku AND <a> -> Akku
5	OR a	Akku OR <a> -> Akku
6	XOR a	Akku XOR <a> -> Akku
7	EQL a	falls Akku = <a>:-1 -> Akku
		sonst : 0 -> Akku
8	JMP a	a -> IAR
9	JMN a	falls Akku < 0 : a -> IAR
F0	HALT	stoppt die MIMA
F1	NOT	bilde Eins-Komplement von Akku -> Akku
F2	RAR	rotiere Akku eins nach rechts -> Akku

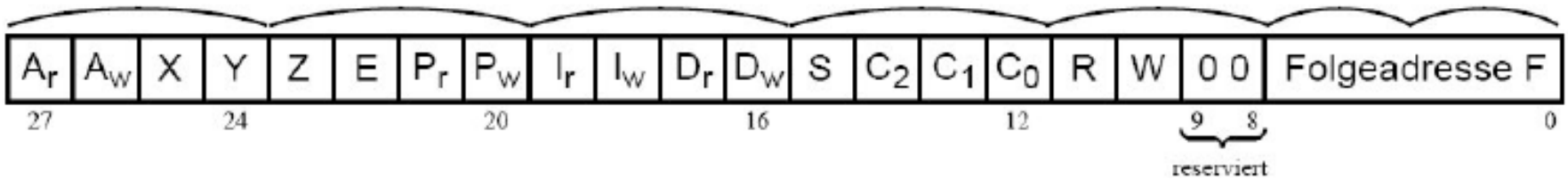
Befehlsformate



ALU-Operationen der MIMA

C_2 C_1 C_0	*ALU Operation*
0 0 0	tue nichts (d.h. Z -> Z)
0 0 1	X + Y -> Z
0 1 0	rotiere X nach rechts -> Z
0 1 1	X AND Y -> Z
1 0 0	X OR Y -> Z
1 0 1	X XOR Y -> Z
1 1 0	Eins-Komplement von X -> Z
1 1 1	falls X = Y, -1 -> Z, sonst 0 -> Z

Mikrobefehlsformat der MIMA



MIMA-Befehl JIND (Jump Indirect)

- JIND a: Neuer Befehl mit dem Opcode D
- `<a>` $\longrightarrow$ `IAR`
- Springt zu der Adresse, die in den 20 niederwertigen Bits der Speicherstelle steht, die durch a referenziert wird
- Wie sieht das Mikroprogramm für die Ausführungsphase aus?

MIMA-Befehl JMS (Jump Subroutine)

- JMS a: Neuer Befehl mit dem Opcode C
- `IAR + 1` $\longrightarrow$ `<a>; a + 1` $\longrightarrow$ `IAR`
- Schreibt die Folgeadresse der aktuellen Adresse an die Speicherstelle a und springt zur Adresse, die auf a folgt
- Wie sieht das Mikroprogramm für die Ausführungsphase aus?

Wochentagsberechnung in MIMA

- Aufgabe: Zu einem Datum in 2004 den Wochentag (0 = Sonntag, 1 = Montag, ...) berechnen
- Vorgegebene Speicherstellen:
  - 0x00020: Tag (Eingabe)
  - 0x00021: Monat (Eingabe)
  - 0x00030: Wochentag (Ausgabe)
  - 0x00100: Startpunkt des Programms
- Algorithmus (in C):

```
int ersterTag[]={-1,4,0,1,4,6,2,4,0,3,5,1,3};
int wochentag = TAG + ersterTag[MONAT] - 1;
wochentag %= 7;
```

Fertig!