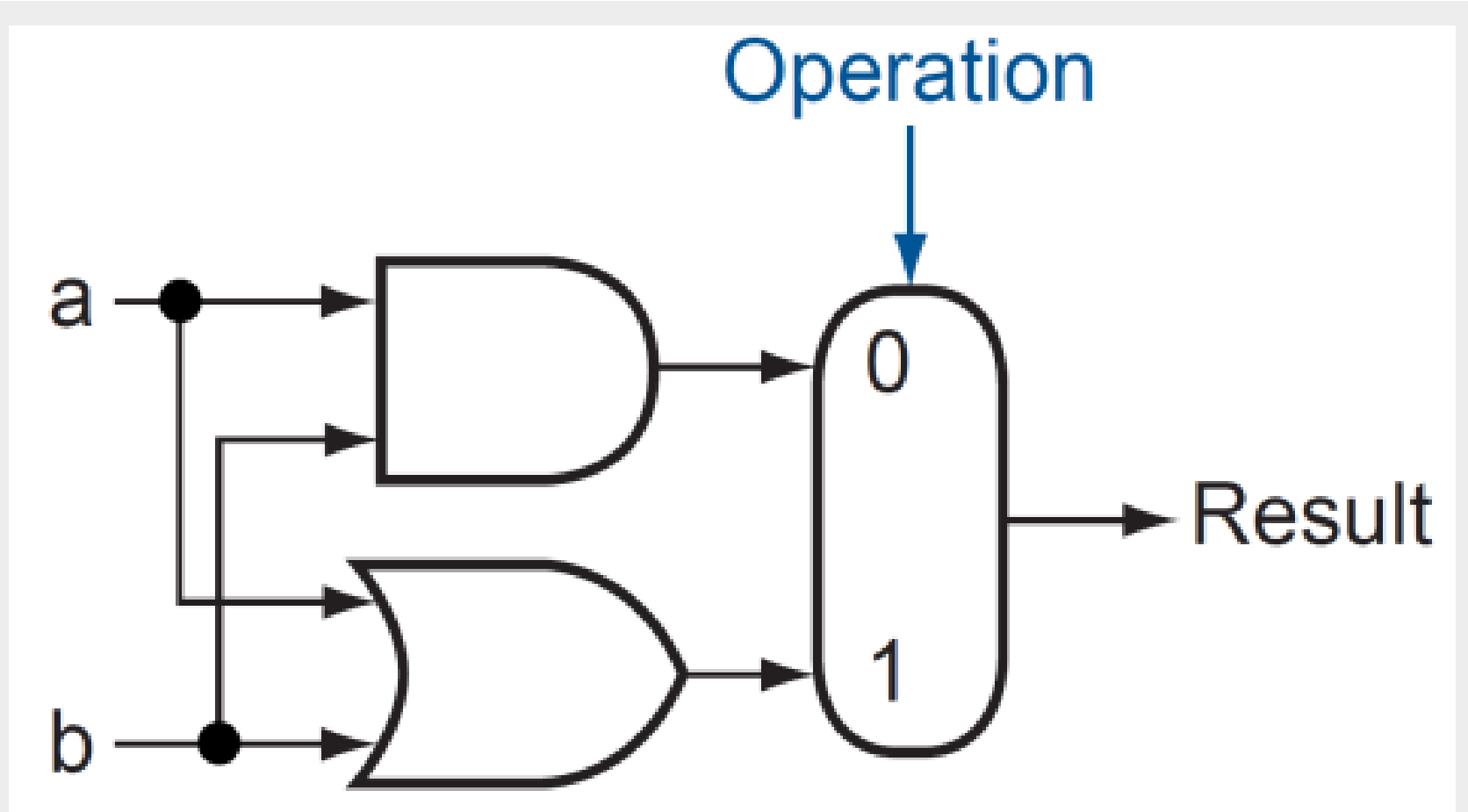
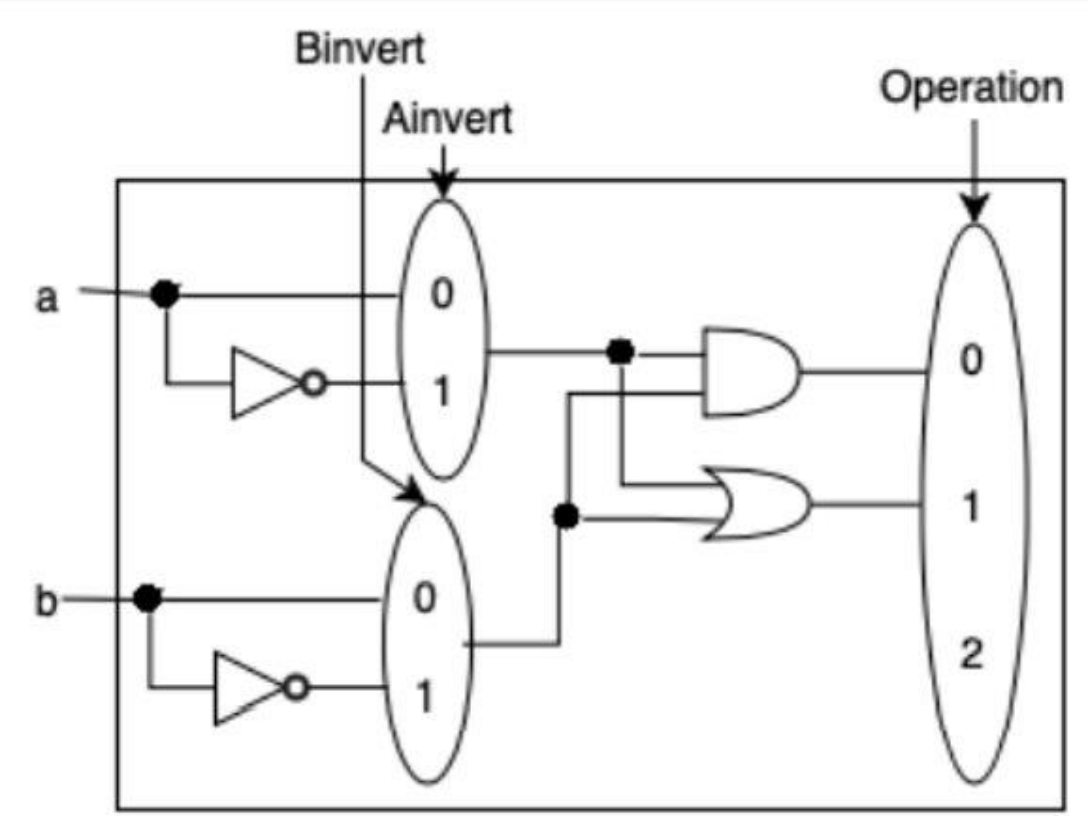
Wiederholung – ALU, KV, PLA

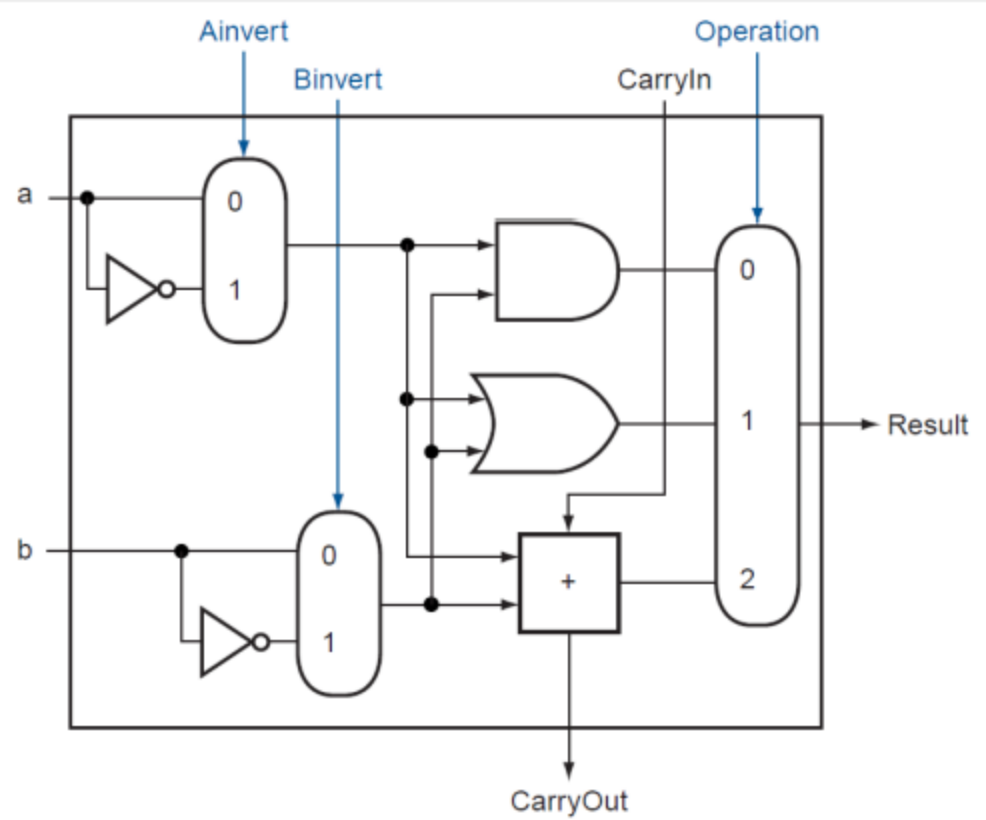
AND & OR



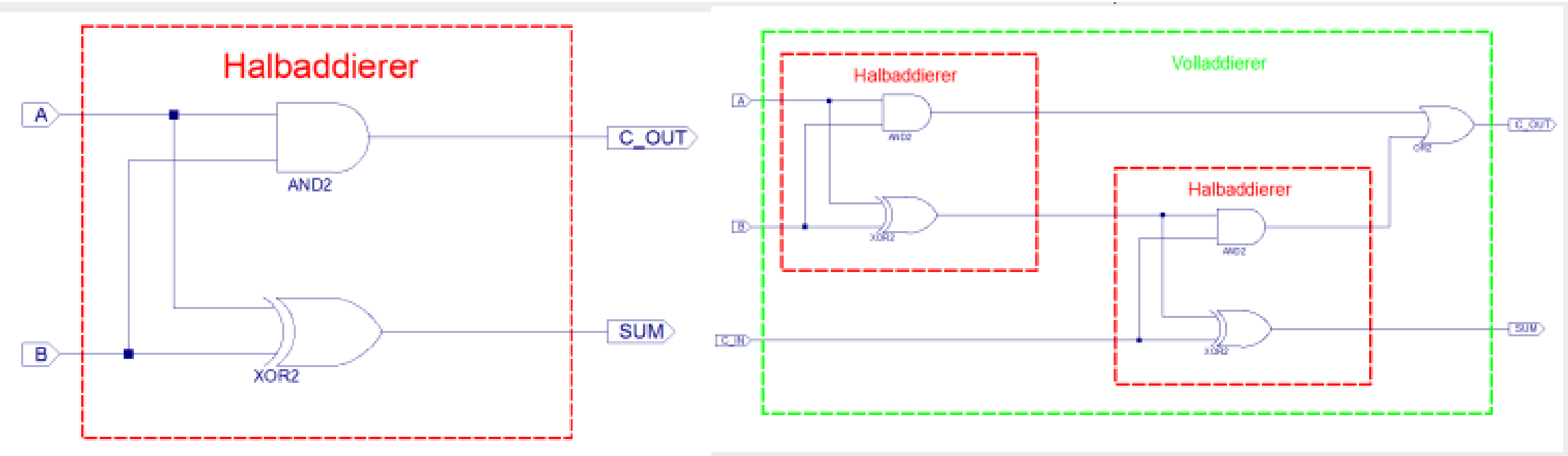
NAND & NOR

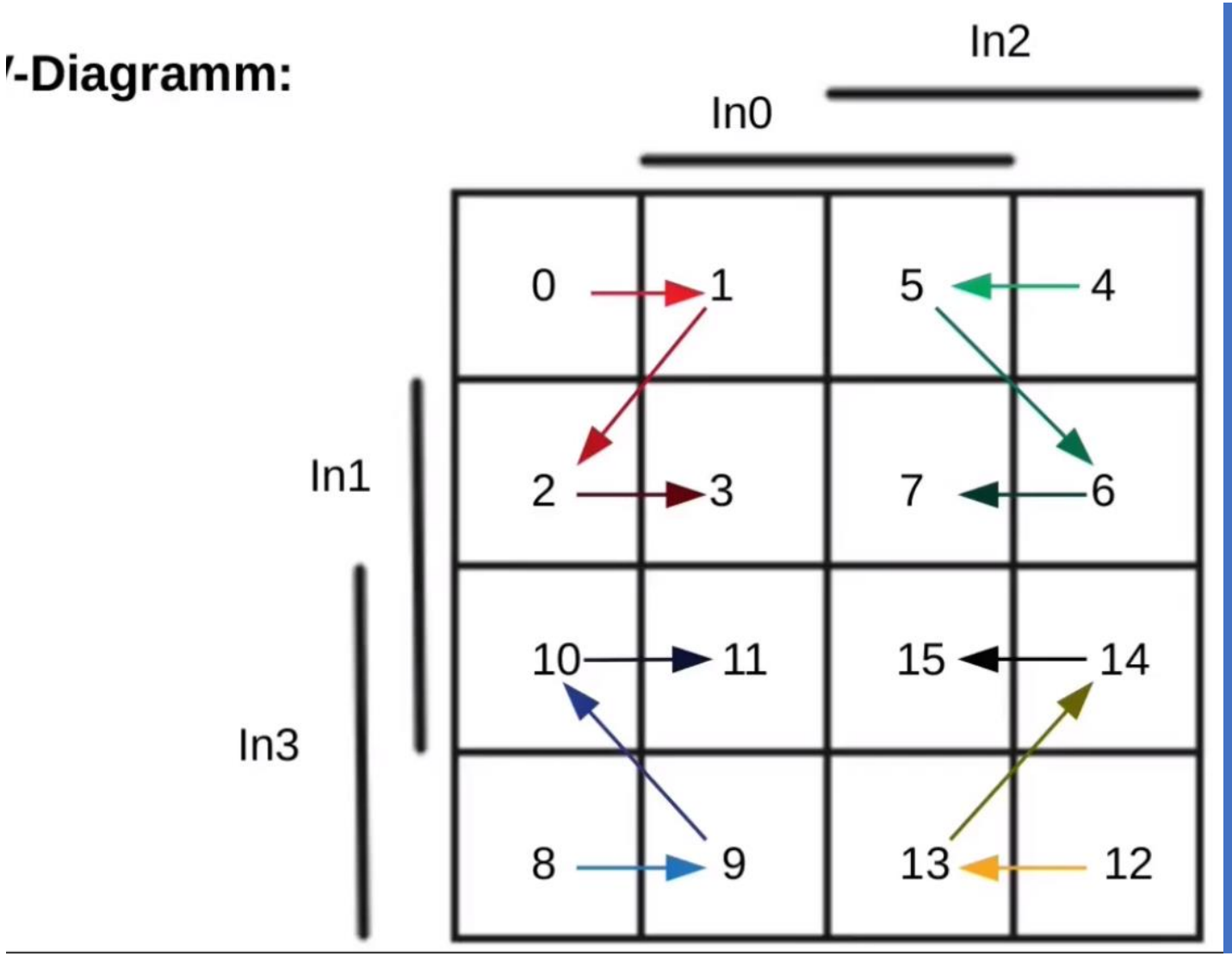


ADD & SUB



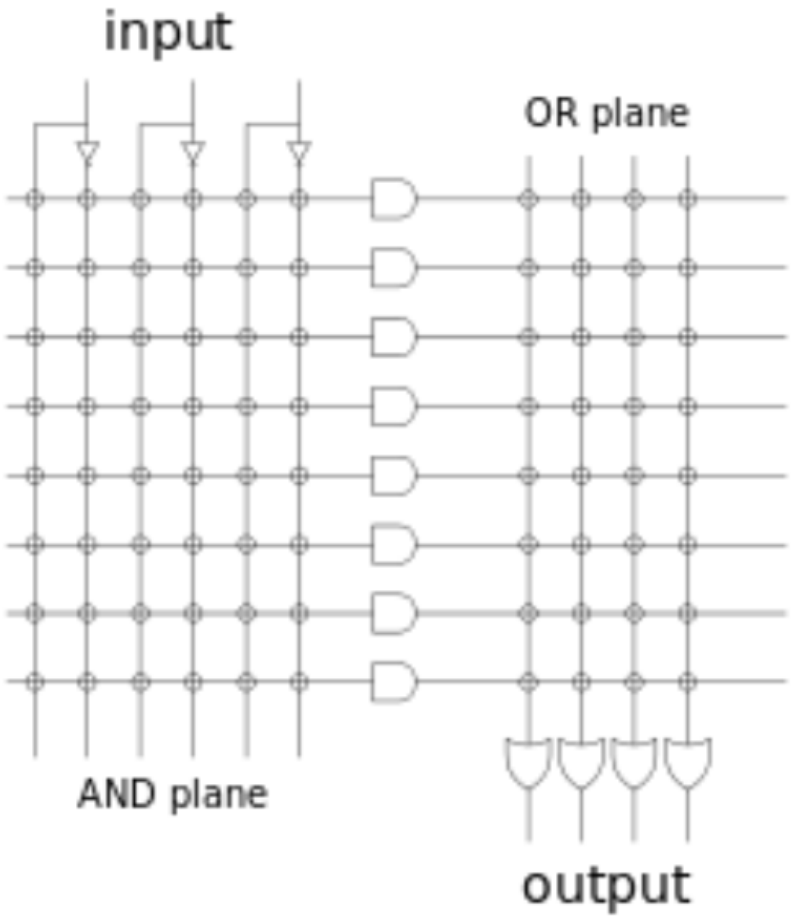
Addition


Halbaddierer
A
B
AND2
XOR2
C_OUT
SUM
Halbaddierer
Volladdierer
Halbaddierer



- Werte für betrachteten Output Eintragen (0 oder 1)
- Menge an Vierecken finden, die:
  - Minimal ist (so wenig wie möglich)
  - Größtmöglich für die einzelnen Vierecke
  - Viereckgröße: 1, 2, 4, 8, 16
  - Alle 1 (und nichts anderes) abdeckt (DNF)
  - Alle 0 (nichts anderes) abdeckt (KNF)

PLA – Für DNF



Programmierbare Logische Anordnung, *Programmable Logic Array (PLA)*

MARS-Programmierung

Register

REGISTER NAME, NUMBER, USE, CALL CONVENTION

NAME	NUMBER	USE	PRESERVED ACROSS A CALL?
$zero	0	The Constant Value 0	N.A.
$at	1	Assembler Temporary	No
$v0-$v1	2-3	Values for Function Results and Expression Evaluation	No
$a0-$a3	4-7	Arguments	No
$t0-$t7	8-15	Temporaries	No
$s0-$s7	16-23	Saved Temporaries	Yes
$t8-$t9	24-25	Temporaries	No
$k0-$k1	26-27	Reserved for OS Kernel	No
$gp	28	Global Pointer	Yes
$sp	29	Stack Pointer	Yes
$fp	30	Frame Pointer	Yes
$ra	31	Return Address	Yes

- Speichern Zahlen (32Bit breit)
- So wie Variablen
- Sehr schneller Speicher, direkt im Prozessor
- Register-Konventionen
  - Bestimmte Weise Register zu verwenden
  - Können gebrochen werden
  - Gibt Punktabzug

Befehle in Assembler-Programmierung

- Sehr simpel
  - Leicht auf Prozessor umzusetzen
  - Höchstens 3 Operanden
  - Begrenzter Befehlssatz
- Stark eingeschränkt
  - Keine While-Schleifen/Loop-Schleifen
    - Durch Sprünge zwischen Befehlen im Programmtext realisiert

Kategorie	Befehl	Beispiel	Bedeutung
Arithmetische Befehle	add	`add $s1,$s2,$s3`	`$s1 = $s2+$s3`
	subtract	`sub $s1,$s2,$s3`	`$s1 = $s2-$s3`
	add immediate	`addi $s1,$s2,100`	`$s1 = $s2+100`
Datentransport	load word	`lw $s1,100($s2)`	`$s1 = Mem[$s2+100]`
	store word	`sw $s1,100($s2)`	`Mem[$s2+100] = $s1`

Komplexe Rechnungen

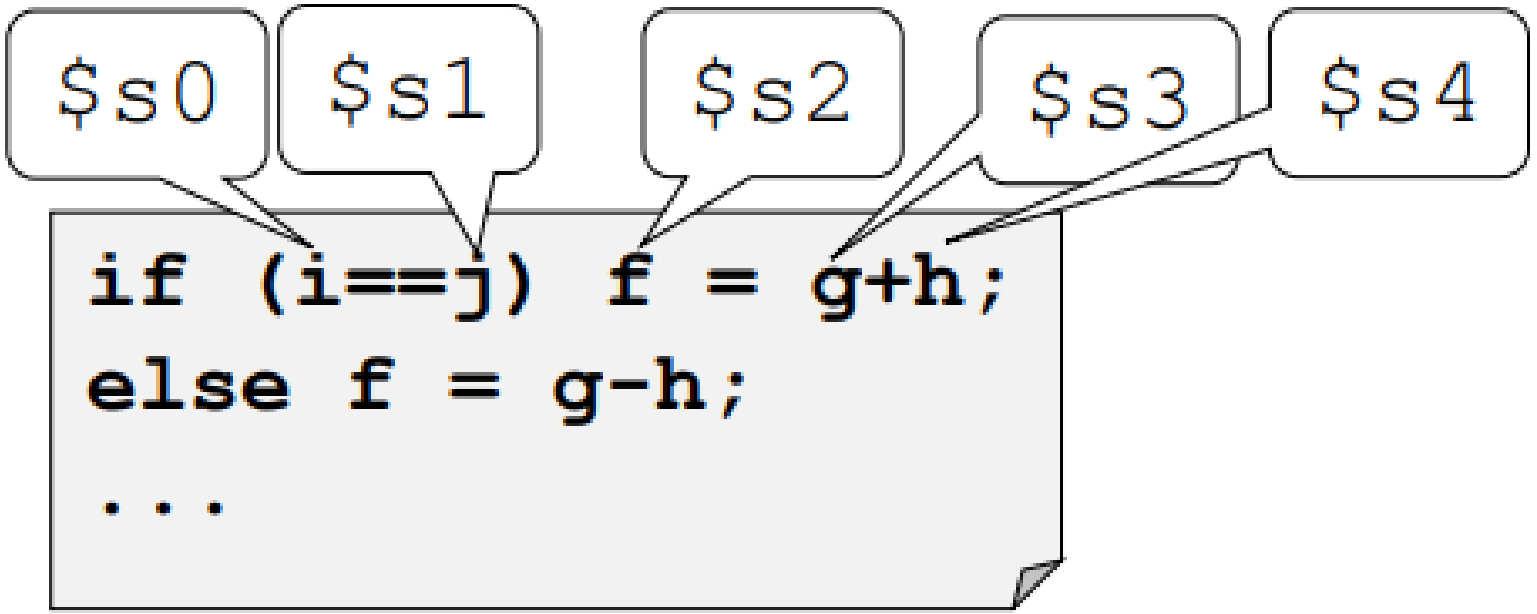
- Schritte müssen aufgeteilt werden
  - Nur 3 Register zur Verwendung

```
f = (g + h) - (i + j);
```

```
add   $t0,$s1,$s2        # $t0 = g+h
add   $t1,$s3,$s4        # $t1 = i+j
sub   $s0,$t0,$t1        # f = $t0-$t1
```

Kontrollflussbefehle

- C/Java:



Kontrollflussbefehle

- MIPS

```
   beq   $s0,$s1,if    # if (i==j) goto if
   sub   $s2,$s3,$s4   # f = g-h (else-part)
   j     endif         # goto endif
if:
   add   $s2,$s3,$s4   # f = g+h (if-part)
endif:
     ...
```