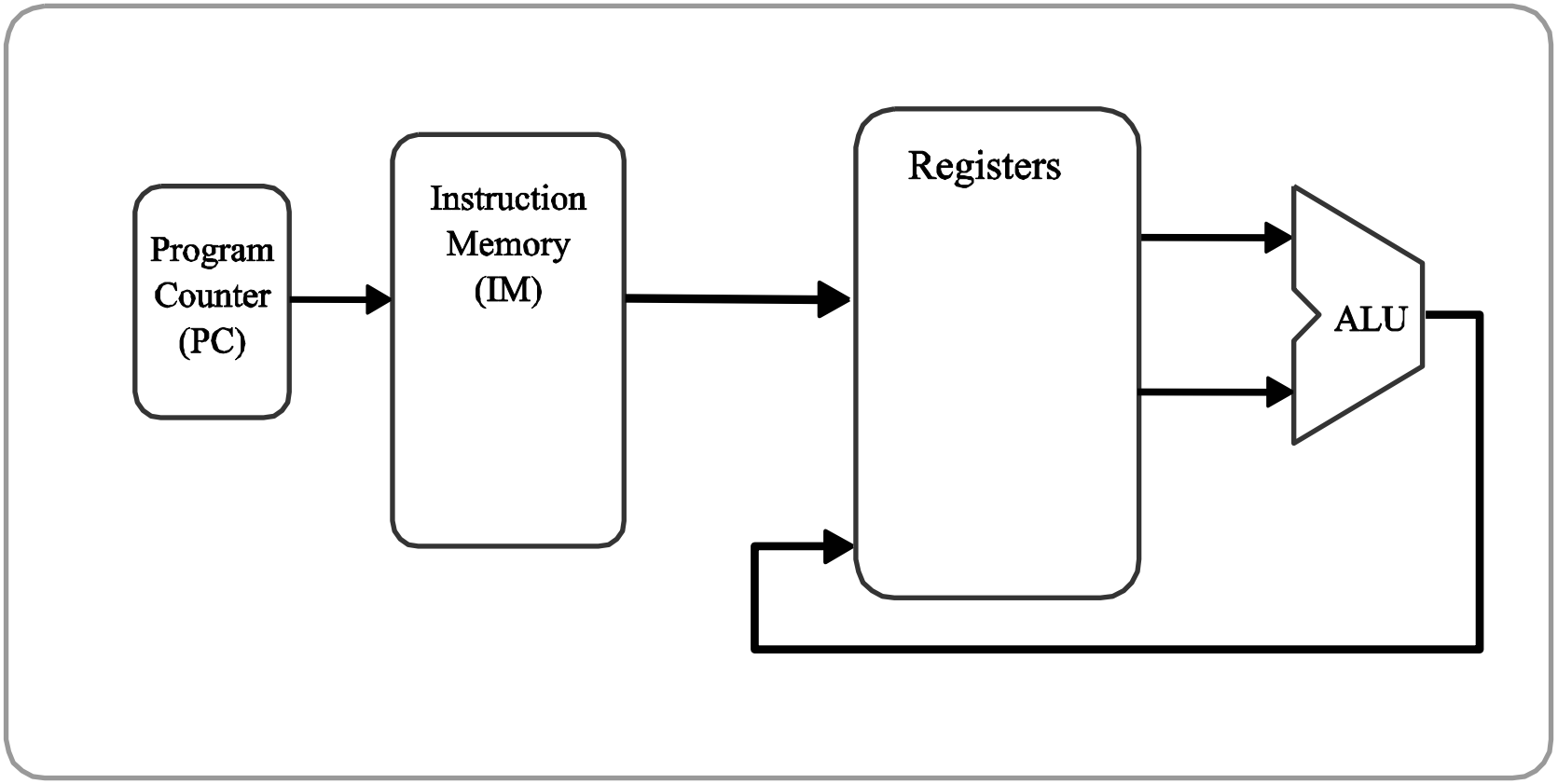
Hoe werkt een “loopje” nu precies?

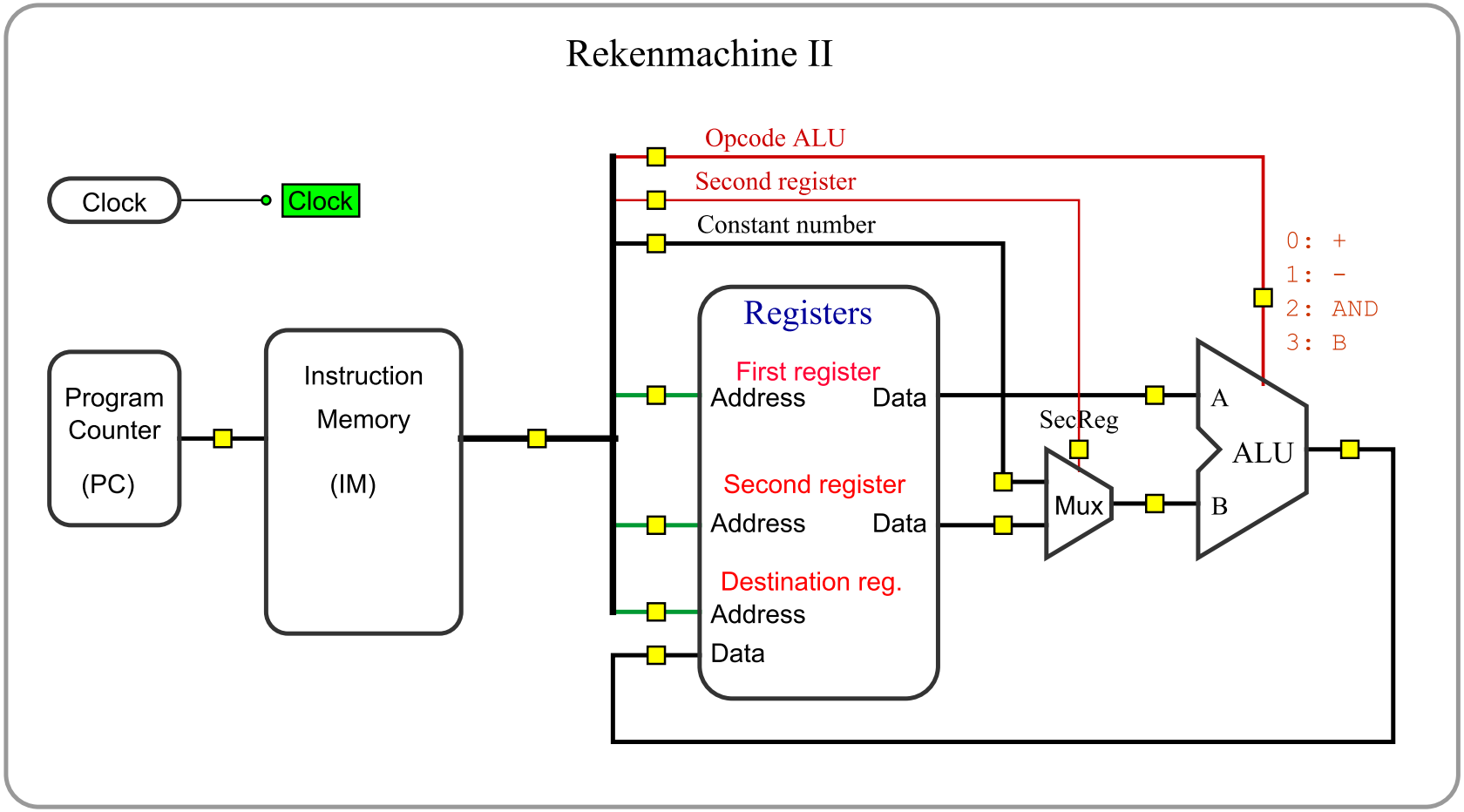
- Recapitulatie rekenmachines hoofdstuk 3
- Van rekenmachine naar rekenmachine met “loopjes”

Recapitulatie rekenmachines

Vier hoofdcomponenten en hun samenhang



Rekenmachine II



Instructieset

- Rekenkundige en logische instructies
  - ADD
  - SUB
  - AND
- Immediate instructies (Instructies met één constant getal)
  - LOADI
  - ADDI
  - SUBI
  - ANDI
- Datatransfer
  - COPY

Instructieset van Rekenmachine II

Instructie	Betekenis	Voorbeeld	Betekenis
ADD rd, rs, rt	Optellen registers	ADD $5, $6, $7	r5 ← r6 + r7
SUB rd, rs, rt	Aftrekken registers	SUB $5, $6, $7	r5 ← r6 - r7
AND rd, rs, rt	Bitwise AND registers	AND $5, $6, $7	r5 ← r6 & r7
COPY rd, rt	Copy register	COPY $3, $2	r3 ← r2
ADDI rd, rs, imm	Optellen register en constante	ADDI $5, $6, 0x1234	r5 ← r6 + 0x1234
SUBI rd, rs, imm	Aftrekken register en constante	SUBI $7, $6, 0x1234	r7 ← r6 - 0x1234
ANDI rd, rs, imm	Bitwise AND register en const.	ANDI $5, $6, 0d34	r5 ← r6 & 0d34
LOADI rd, imm	Laad getal in register	LOAD $1, 0x 0020	r1 ← 0x0020

Hoe werkt een “loopje” nu precies?

- Recapitulatie rekenmachine
- Van rekenmachine naar rekenmachine met “loopjes”

Uitbreiding instructieset met Branch-instructies:

- Rekenkundige en logische instructies
  - ADD
  - SUB
  - AND
  - COPY
- Immediate instructies (Instucties met één constant getal)
  - ADDI
  - SUBI
  - ANDI
  - LOADI
- Branch instucties
  - BZ (Branch if zero)
  - BEQ (Branch if equal)
  - BRA (Branch always)

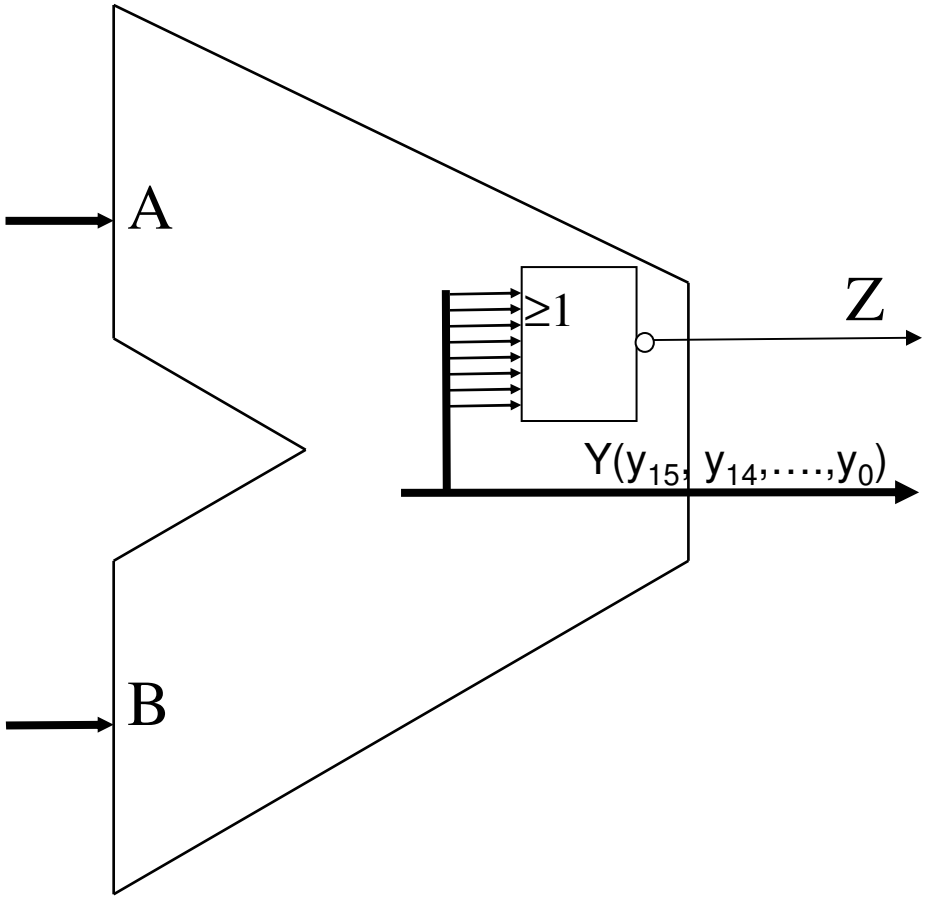
Voorwaarden Branch-instructie

Een sprong in een programma wordt alleen uitgevoerd als:
- er een Branch-instructie geprogrammeerd is én
- de Zero-uitgang van de ALU 1 is.

Uitbreiding hardware

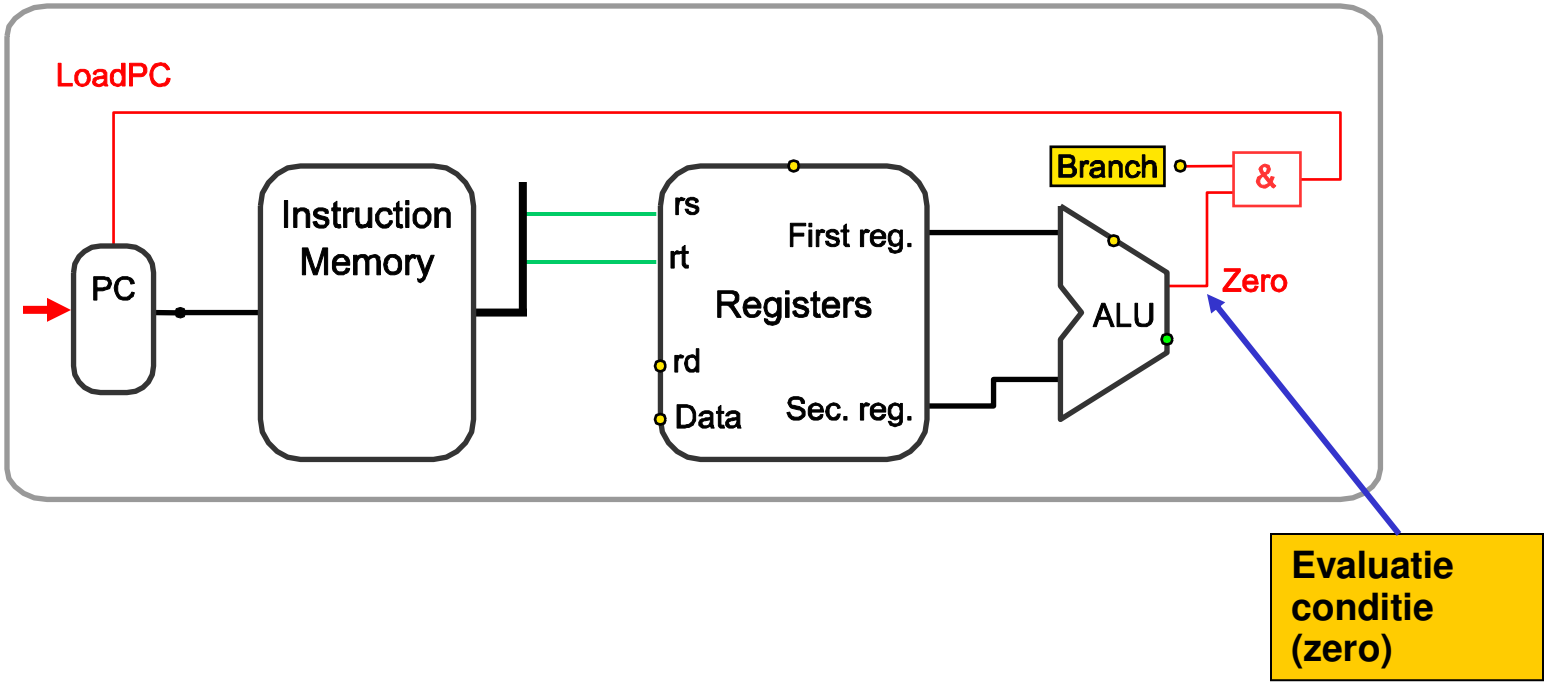
- Componenten die de voorwaarden scheppen om een sprong mogelijk te maken. Dit zijn:
  - een Branch-bit;
  - een extra ALU-uitgang; (zero flag)
  - een AND-poort;
  - een LoadPC-ingang zodat de PC naar een andere waarde kan gaan de volgende;
  - een extra Adder om het adres te berekenen waar de PC naar toe moet springen.

ALU met Zero-flag

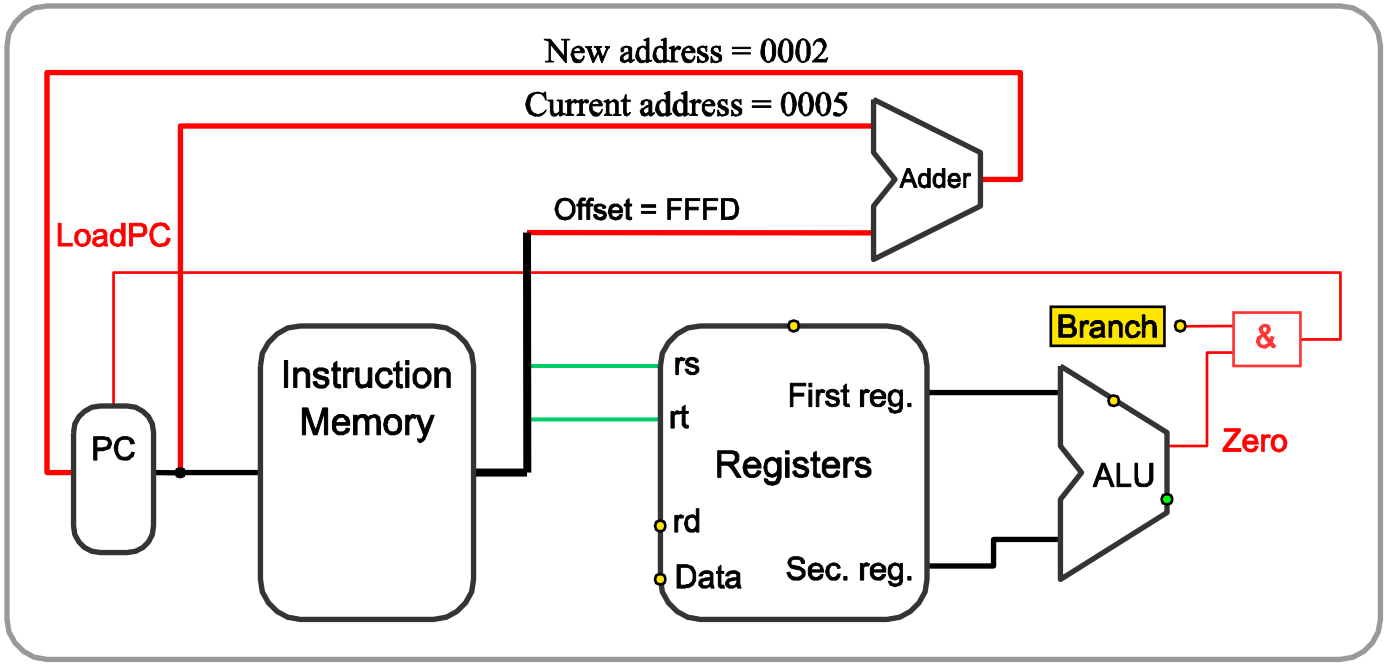


$$Z = \overline{y_{15} + y_{14} + y_{13} + ... + y_0}.$$

Wanneer wordt een branch-instructie uitgevoerd?

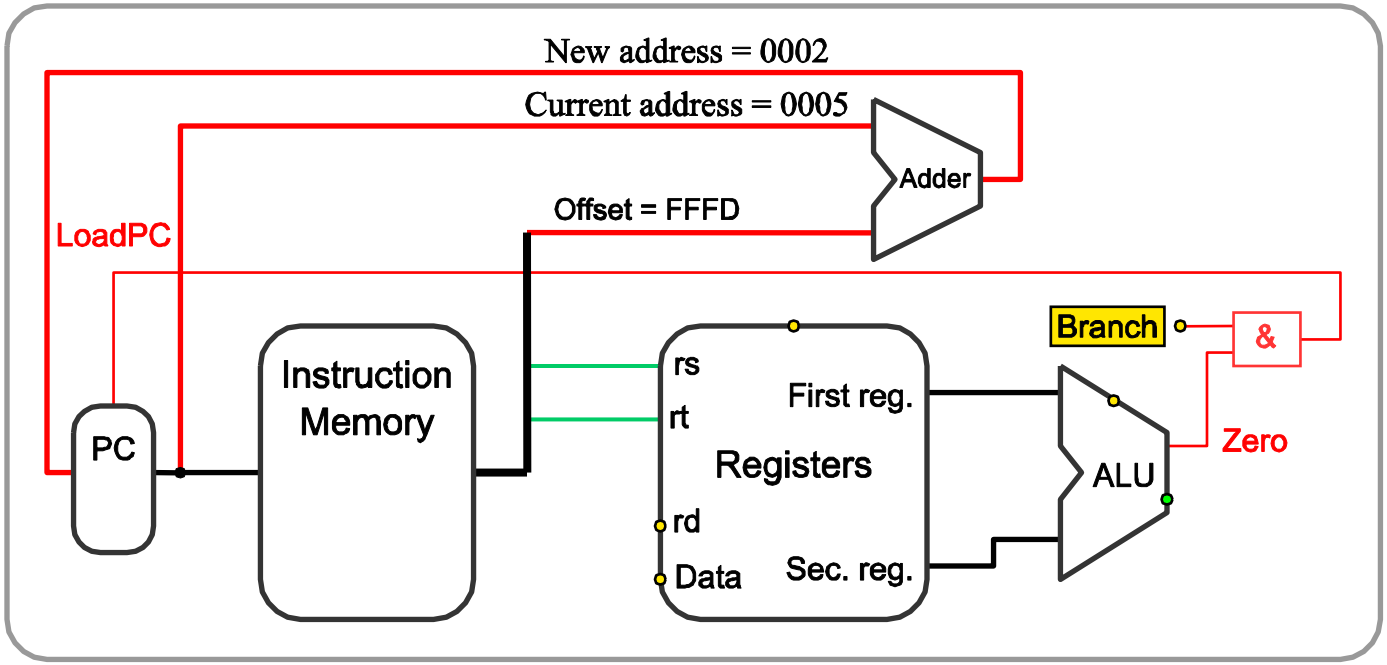


Naar welke instructie springt de PC?



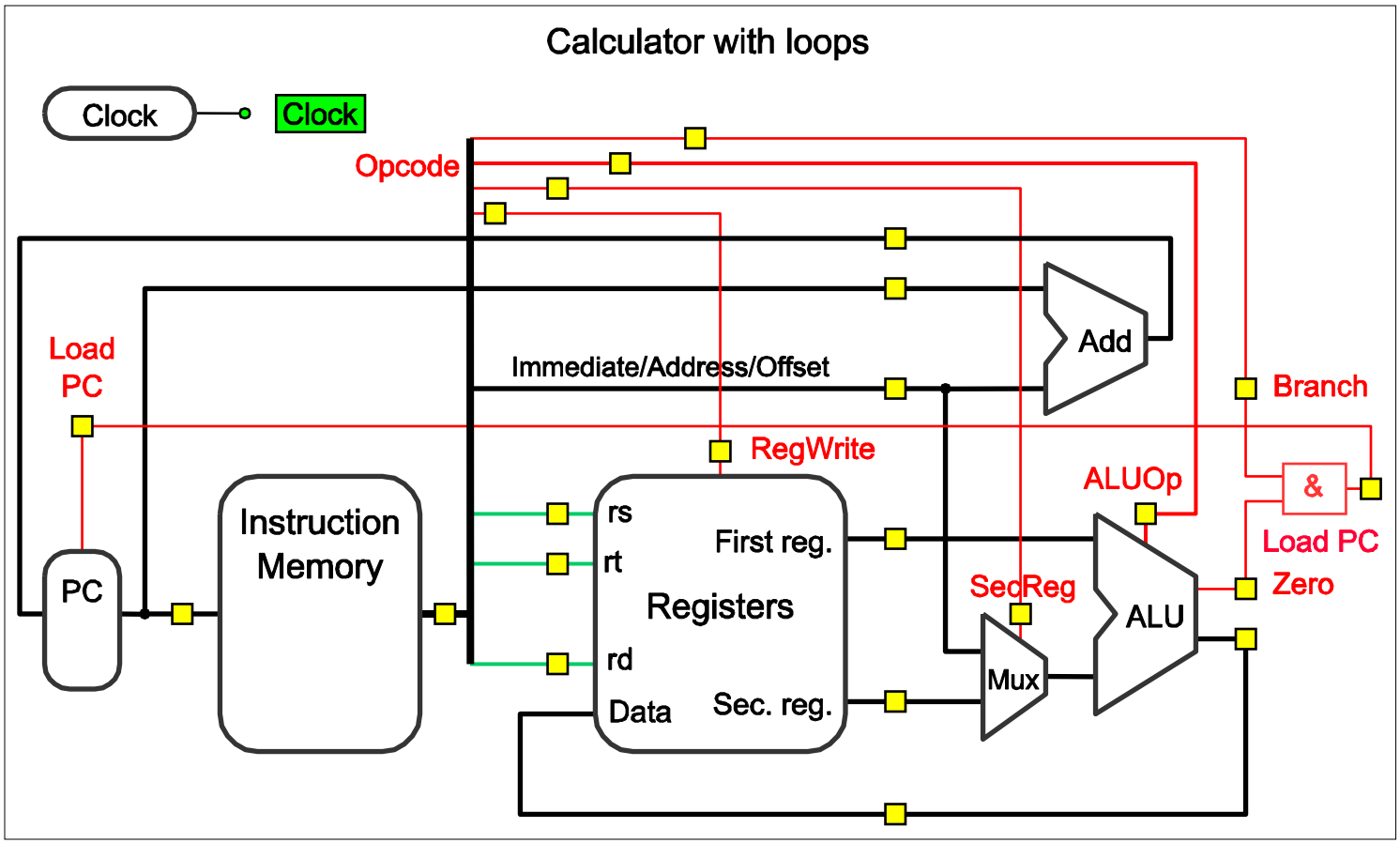
Huidige waarde PC is 5;
FFFD = -3; 3 instructies terug;
Nieuwe waarde PC wordt 2

Naar welke instructie springt de PC?



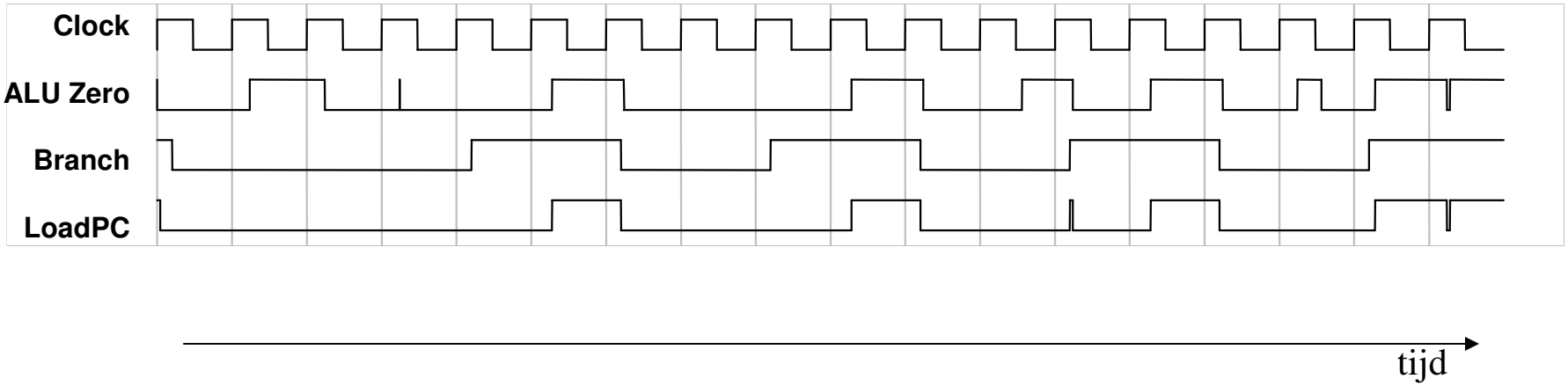
Syntax : BEQ rs, rt, label
Voorbeeld: BEQ $7, $5, label
Betekenis: if (inhoud register 7 = inhoud register 5) goto label

Welke operatie moet de ALU uitvoeren?



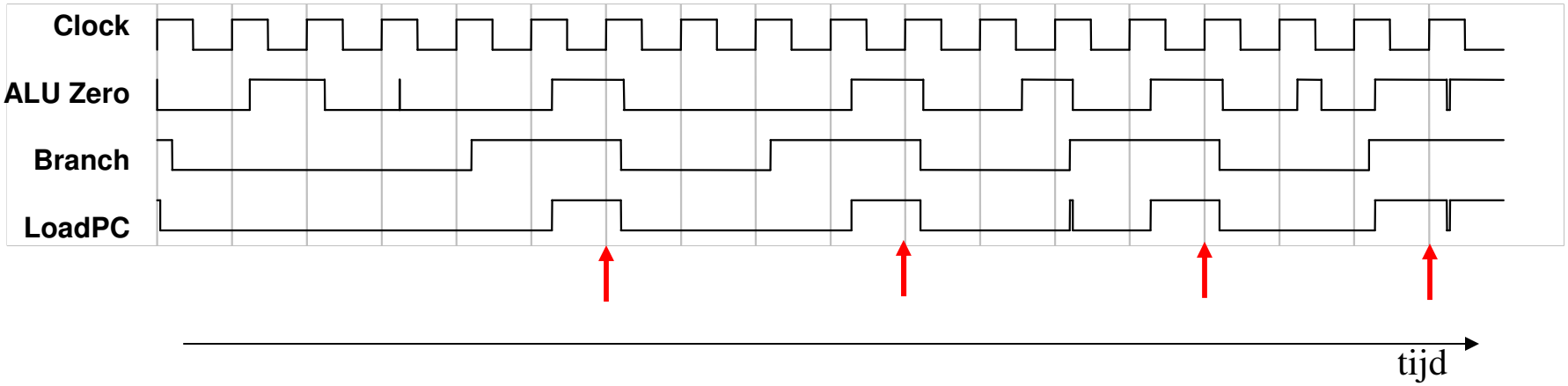
Waarom is er een RegWrite-lijn nodig?

Wanneer wordt er gesprongen?



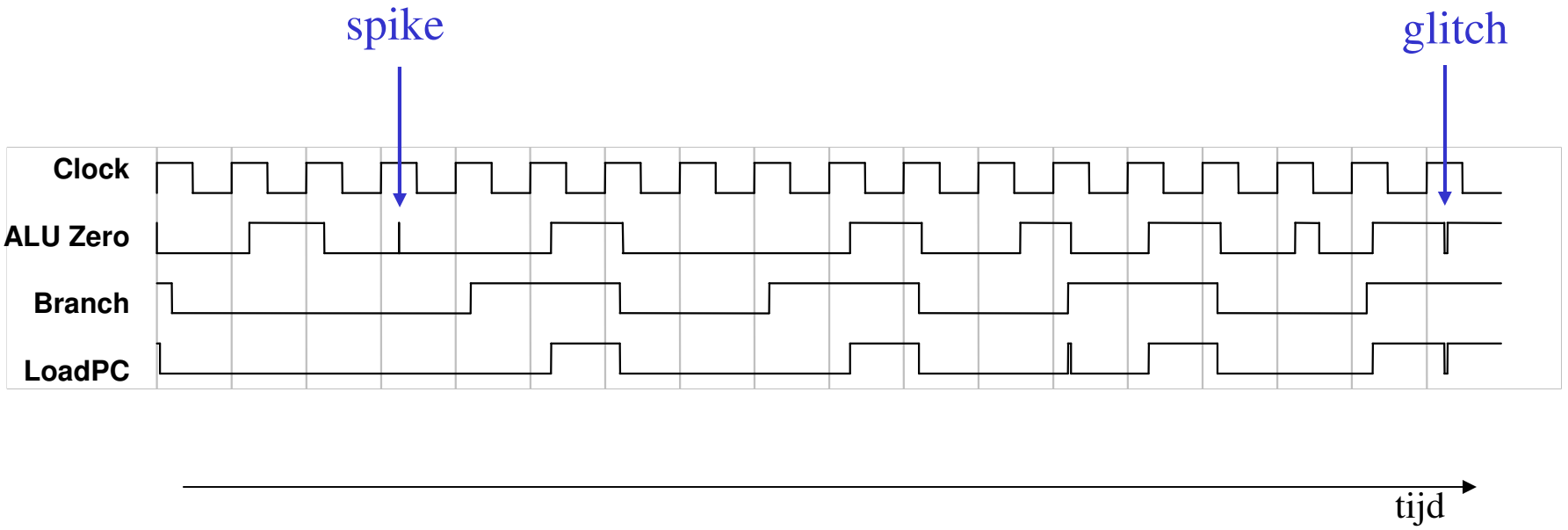
Voorwaarden sprong:
- Branch = 1
- Zero = 1
- Opgaande klokflank

Wanneer wordt er gesprongen?



Voorwaarden sprong:
- Branch = 1
- Zero = 1
- Opgaande klokflank

Spikes & glitches



Toelichting opdracht 8

Vermenigvuldigen op de basisschool

Multiplicand a	1210
Multiplier b	1301
	1210
	0000
	3630
	1210
Product	1574210

Vermenigvuldigen binair

Multiplicand a	1010
Multiplier b	1101
	1010
	0000
	1010
	1010
Product	10000010

Vermenigvuldigen binair

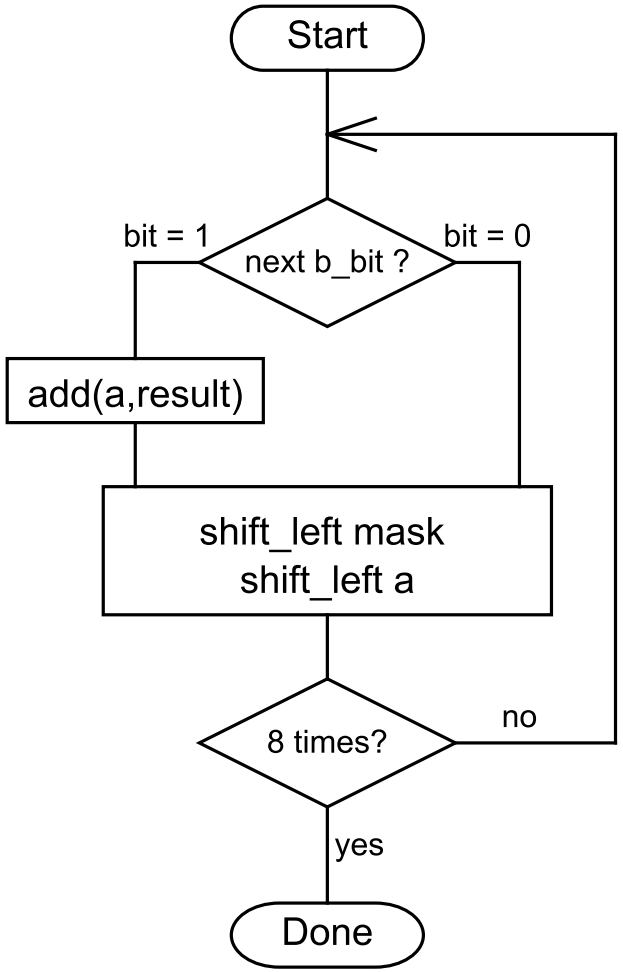
Multiplicand a	1010
Multiplier b	1101
	1010
	0000
	01010
	1010
	110010
	1010
Product	10000010

Vermenigvuldigen binair

Multiplicand a	1010
Multiplier b	1101
	1010
	1010
	110010
	1010
Product	10000010

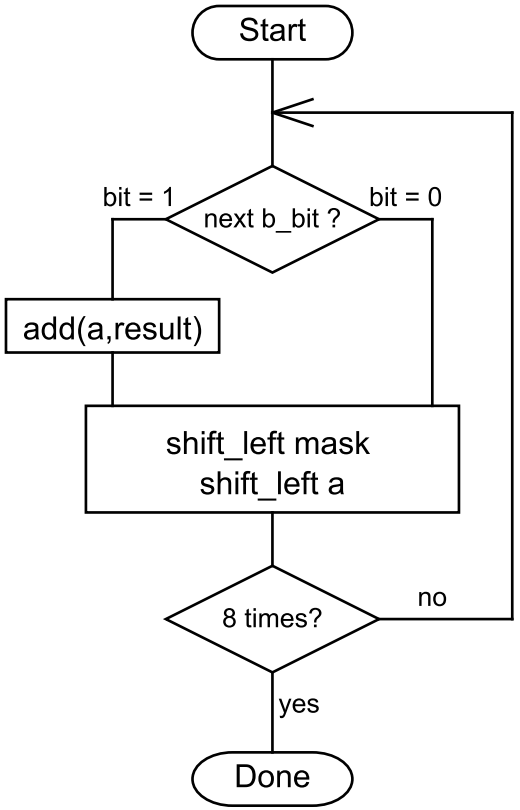
Vermenigvuldigen

Multiplicand a	1010
Multiplier b	1101
	1010
	1010
	110010
	1010
Product	10000010



Shifting & Masking

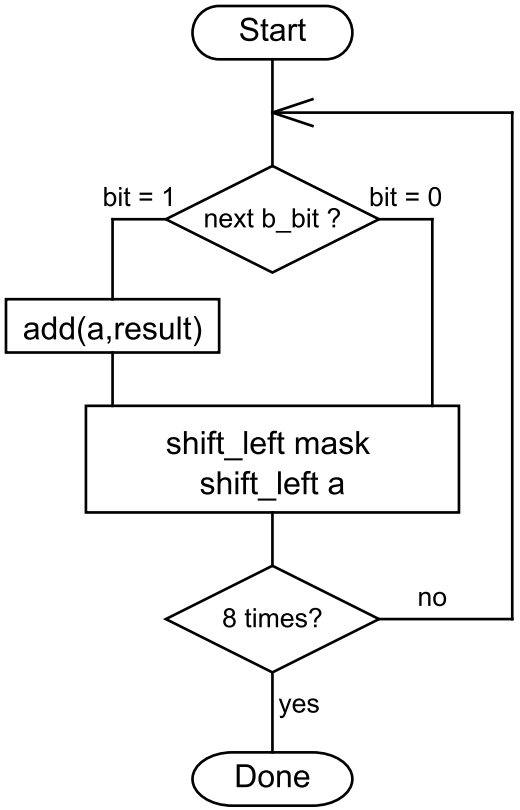
Multiplicand a	1010
Multiplier b	1101
	1010
	1010
	110010
	1010
Product	10000010



b	1101	result	0000
mask	0001	a	1010
b AND mask	0001	ADD(a, result)	1010

Shifting & Masking

Multiplicand a	1010
Multiplier b	1101
	1010
	1010
	110010
	1010
Product	10000010



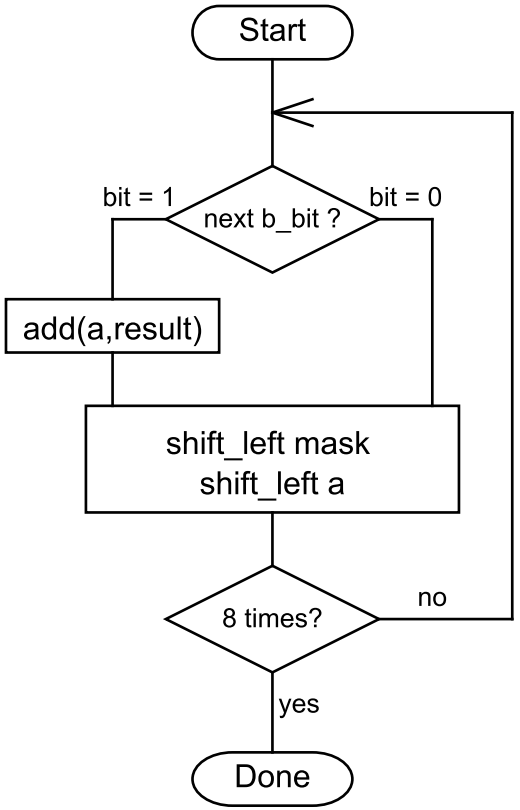
b	1101	result	1010
mask	0010	a	10100
b AND mask	0000	No ADD	

ADD mask, mask, mask = Shl 1, mask

ADD a, a, a = Shl 1, a

Shifting & Masking

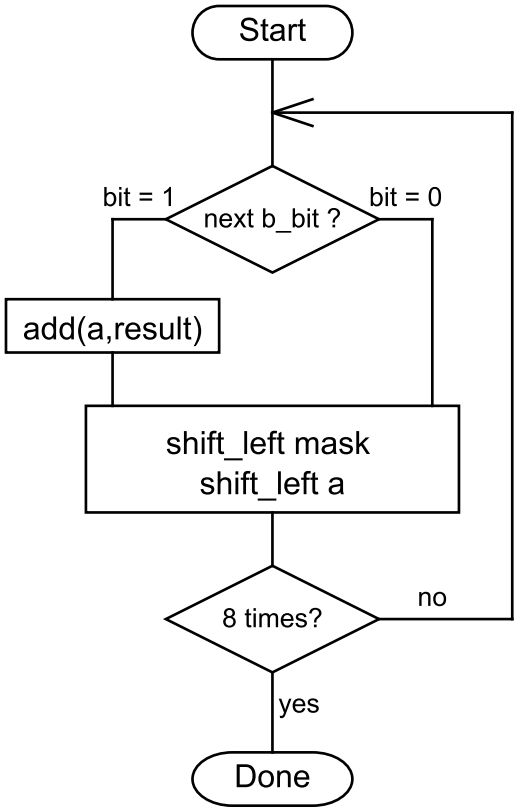
Multiplicand a	1010
Multiplier b	1101
	1010
	1010
	110010
	1010
Product	10000010



b	1101	result	1010
mask	0100	a	101000
b AND mask	0100	ADD(a, result)	110010

Shifting & Masking

Multiplicand a	1010
Multiplier b	1101
	1010
	1010
	110010
	1010
Product	10000010



b	1101	result	110010
mask	1000	a	1010000
b AND mask	1000	ADD(a, result)	10000010