Úvod do organizace počítače

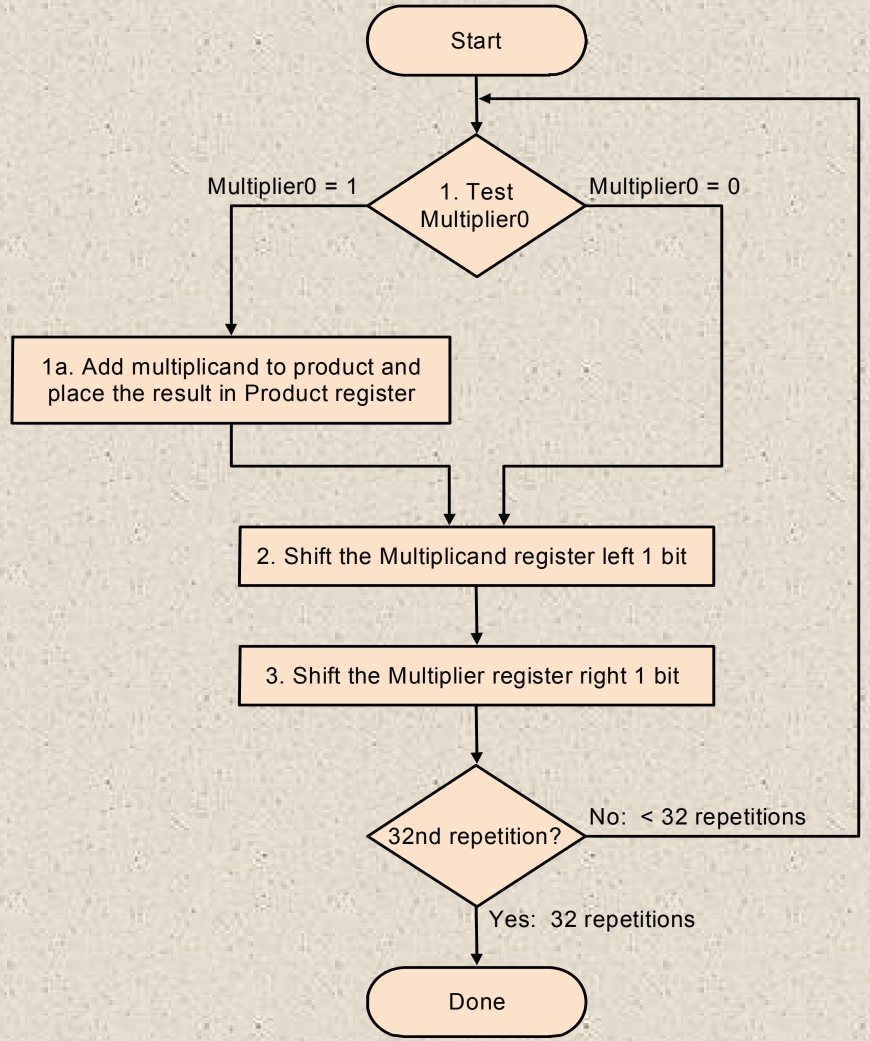
Operace násobení a dělení

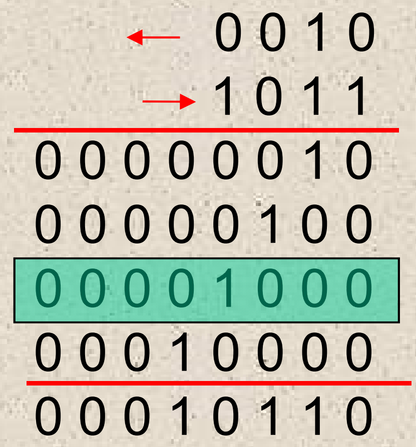
Násobení

- Algoritmus je podobný algoritmu ze základní školy
- Složitější operace než sčítání
  - Realizace pomocí posuvů a sčítání
  - Vyžaduje více času a zabere větší plochu čipu
- 3 verze *algoritmu tužka-a-papír*

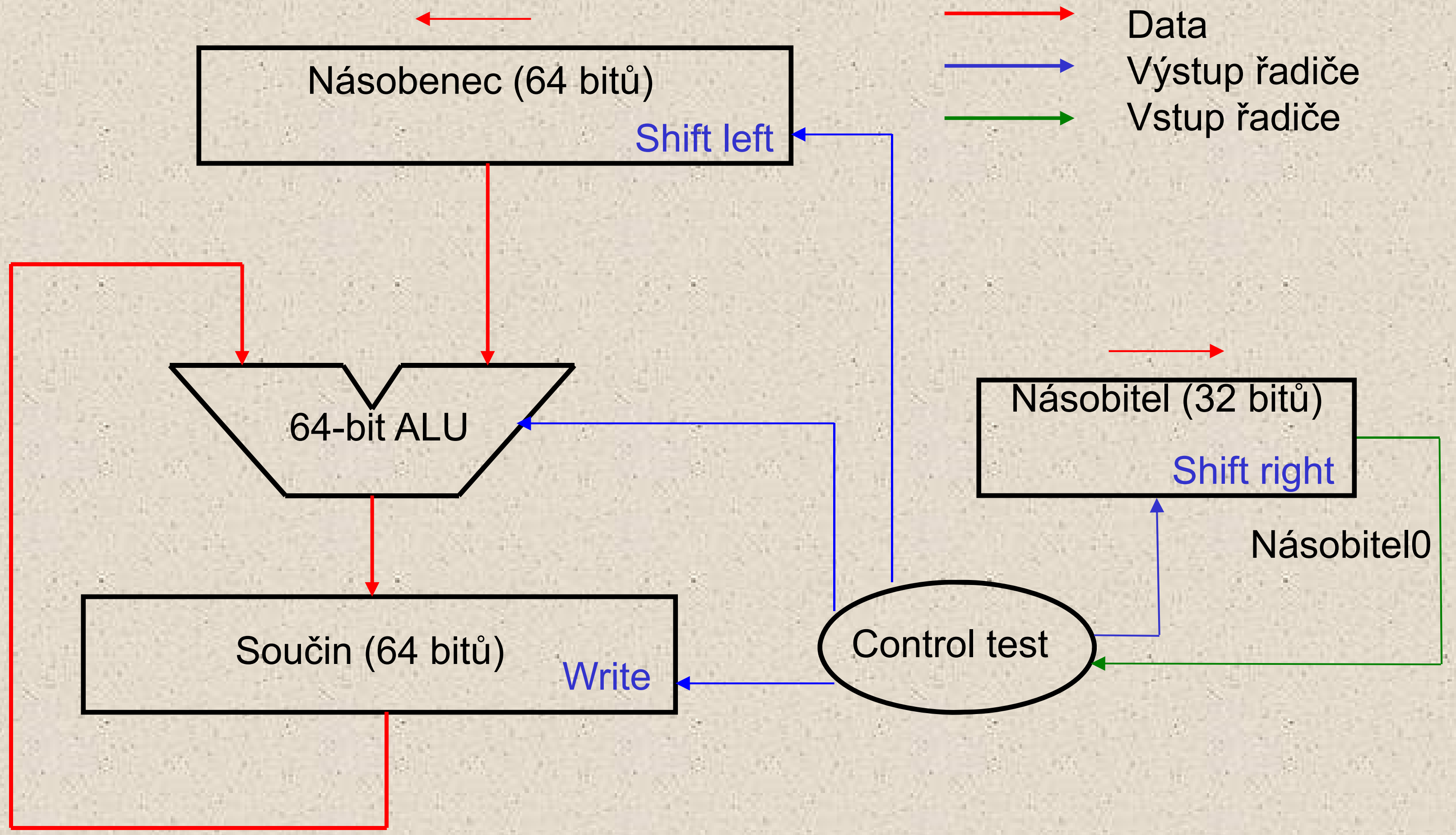
```
       0010   (násobenec)
   __x_1011   (násobitel)
       0010                1 -> copy & shift
      0010                 1 -> copy & shift
     0000                  0 -> shift
    0010                   1 -> copy & shift
  ___________
   00010110                Suma parciálních součinů
```

První verze (V.1)

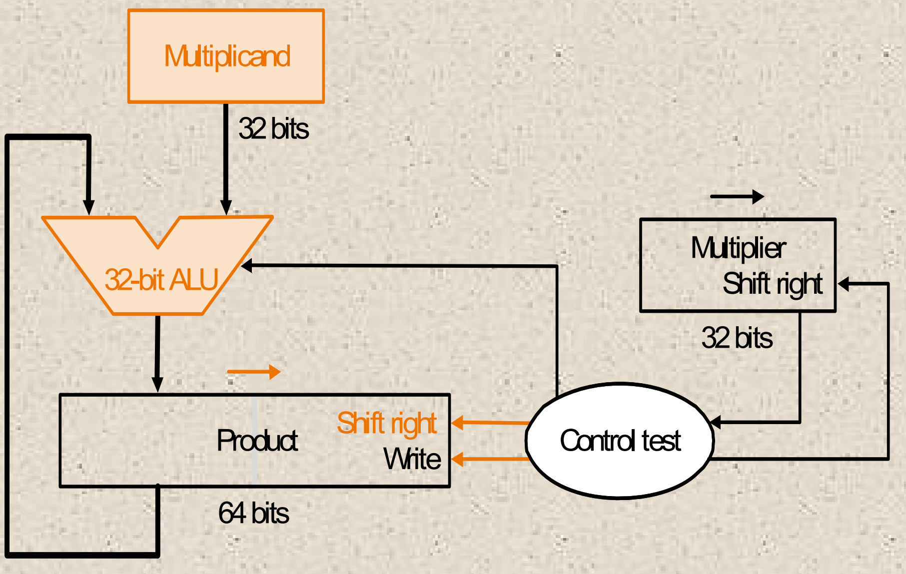


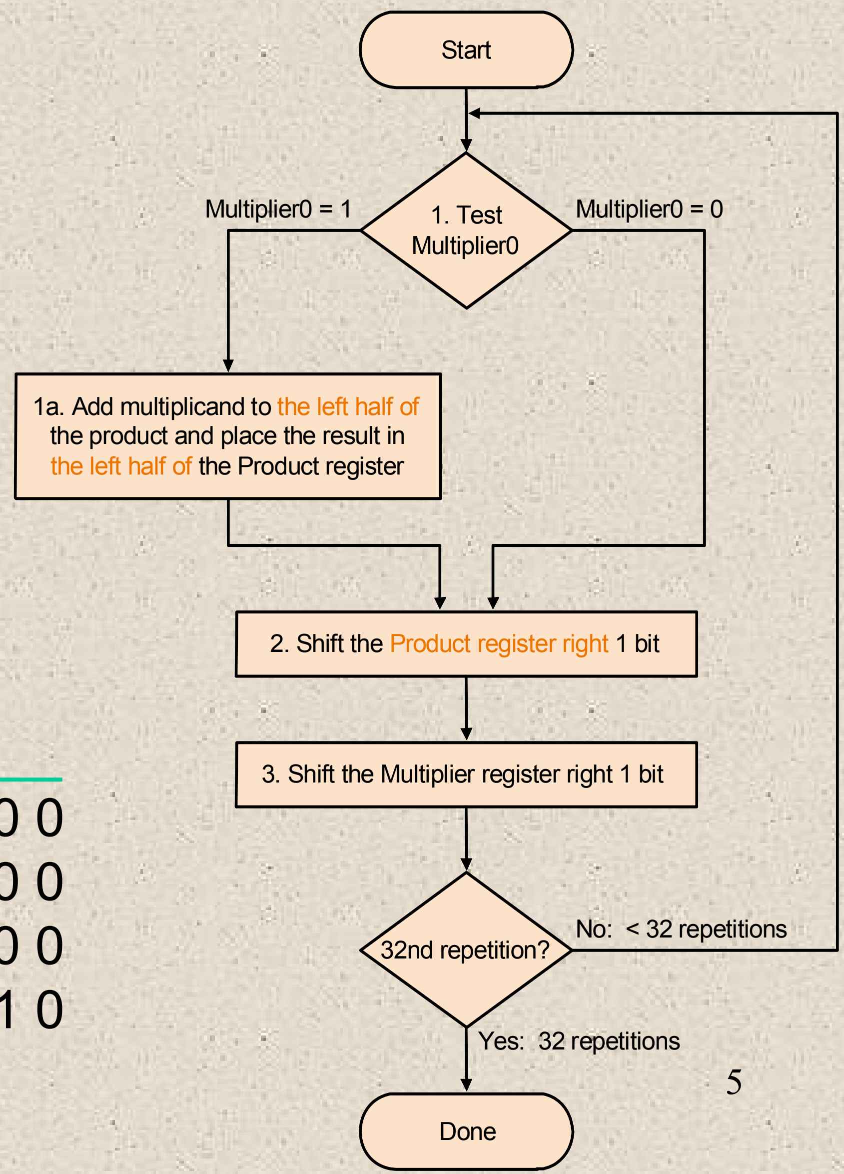


Hardware (V.1)


Data
Výstup řadiče
Vstup řadiče
Násobenec (64 bitů)
Shift left
64-bit ALU
Násobitel (32 bitů)
Shift right
Násobitel0
Součin (64 bitů)
Write
Control test

Druhá verze (V.2)

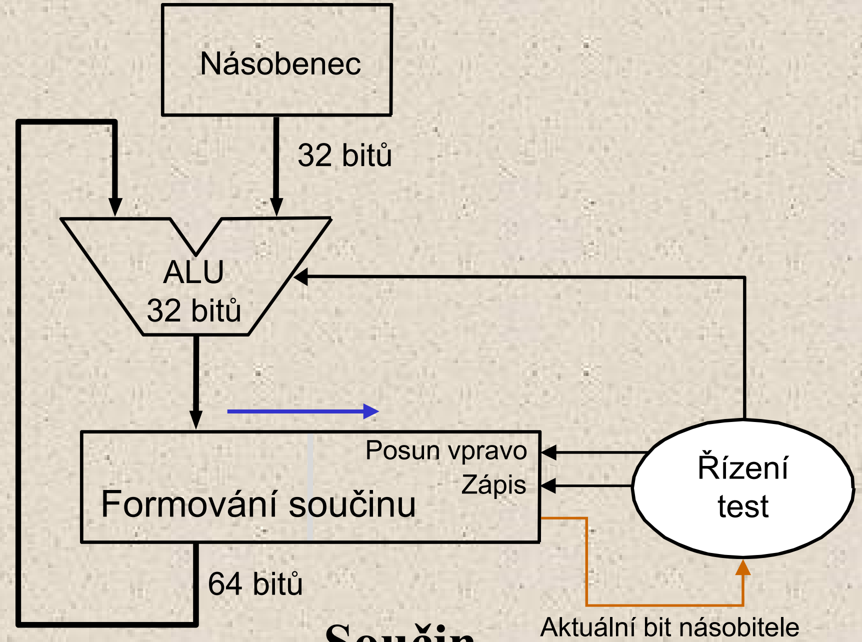


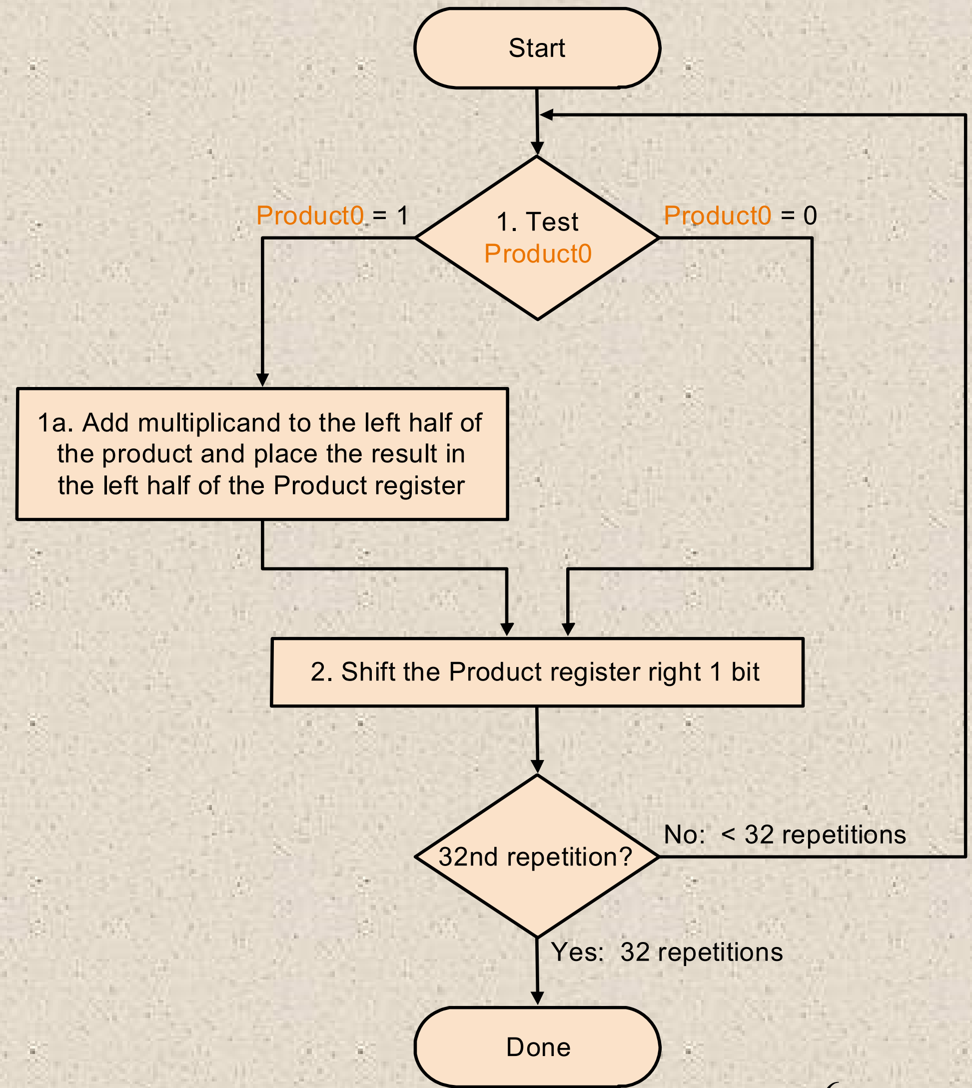


Součin

Násobitel0			
	0 0 0 0 0 0 0 0		
1	0 0 1 0 0 0 0 0	→	0 0 0 1 0 0 0 0
1	0 0 1 1 0 0 0 0	→	0 0 0 1 1 0 0 0
0	0 0 0 1 1 0 0 0	→	0 0 0 0 1 1 0 0
1	0 0 1 0 1 1 0 0	→	0 0 0 1 0 1 1 0

Konečná verze (V.3)





Součin

0 0 0 0 1 0 1 1

0 0 1 0 1 0 1 **1** → 0 0 0 1 0 1 0 1

0 0 1 1 0 1 0 **1** → 0 0 0 1 1 0 1 0

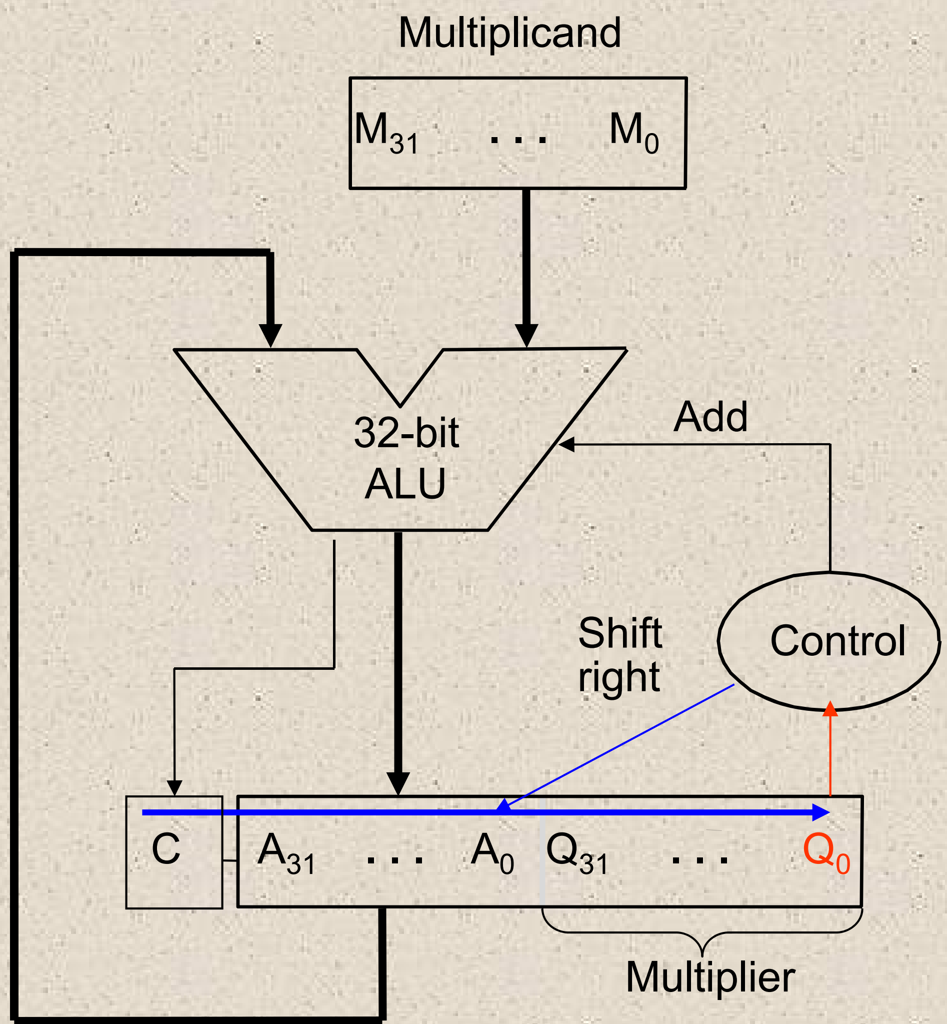
0 0 0 1 1 0 1 **0** → 0 0 0 0 1 1 0 1

0 0 1 0 1 1 0 **1** → 0 0 0 1 0 1 1 0

Souhrn

- Násobení bez znaménka
  - Generace parciálního součinu pro každou cifru násobitele
  - Parciální součin = $\begin{cases} \textbf{0} & \textbf{If cifra násobitele = 0} \\ \textbf{Násobenec} & \textbf{If cifra násobitele = 1} \end{cases}$
  - Celkový součin = součet parciálních součinů (správně posunutých vlevo)
  - Násobení dvou n-bitových binárních čísel dává výsledek o šířce 2n bitů

Celkový pohled



1011 Multiplicand (11)

x 1101 Multiplier (13)

Product

(143)

		C	A	Q	M
	Initial values	0	0000	1101	1011
1	Add	0	1011	1101	1011
	Shift	0	0101	1110	1011
2	Shift	0	0010	1111	1011
3	Add	0	1101	1111	1011
	Shift	0	0110	1111	1011
4	Add	1	0001	1111	1011
	Shift	0	1000	1111	1011

Aritmetika se znaménkem

- Sčítání a odčítání se znaménkem
  - S operandy se zachází jako s čísly bez znaménka
  - Používá stejné algoritmy i hardware upoužívané pro odpovídající operace bez znaménka

	Unsigned	Signed
1 0 0 1	9	-7
+ 0 0 1 1	3	3
1 1 0 0	12	-4

- Pro násobení nelze použít!

Příklad

	Unsigned	Signed
1011	11	-5
x 1101	13	-3
10001111	143	~~-113~~

- Částečné řešení, je-li násobenec záporný

1001	(9)		1001	(-7)	
x 0011	(3)		x 0011	(3)	
00001001	1001×2^0		11111001	$(-7) \times 2^0 =$	(-7)
00010010	1001×2^1		11110010	$(-7) \times 2^1 =$	(-14)
00011011	(27)		11101011	(-21)	

- Nelze použít, je-li násobitel záporný

Záporný násobitel

- Bity násobitele nekorespondují s parciálními součiny
- Příklad: (-3) = 1101
  - Parciální součiny by se generovaly podle stavu bitů násobitele:

    1: -1×2^0

    0: -0×2^1

    1: -1×2^2

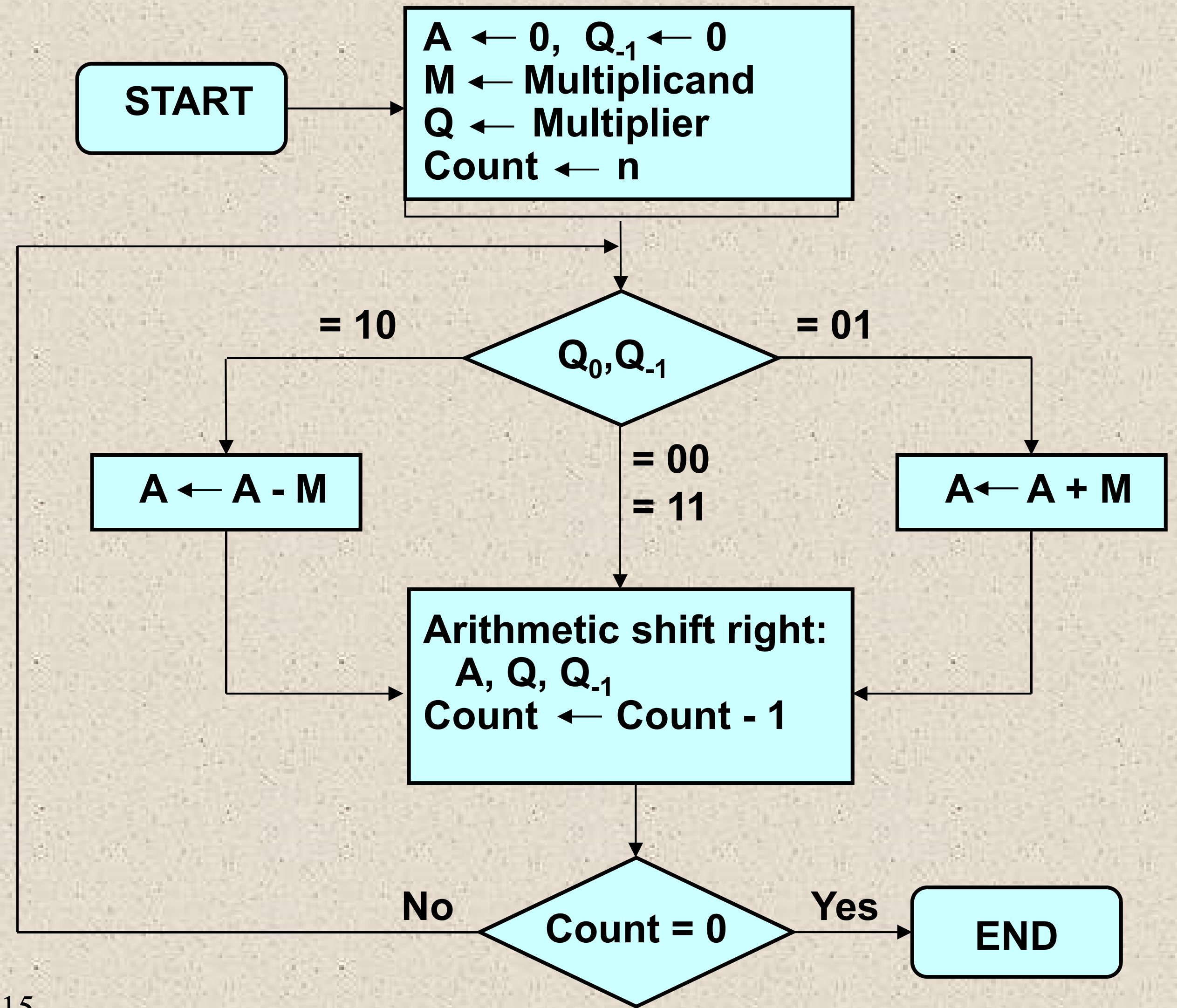
    1: -1×2^3

  - Měly by však být generovány s použitím následujících mocnin 2:

    -1×2^0

    -1×2^1

Boothův algoritmus



Boothův algoritmus

Boothův algoritmus pro dvojkový komplement

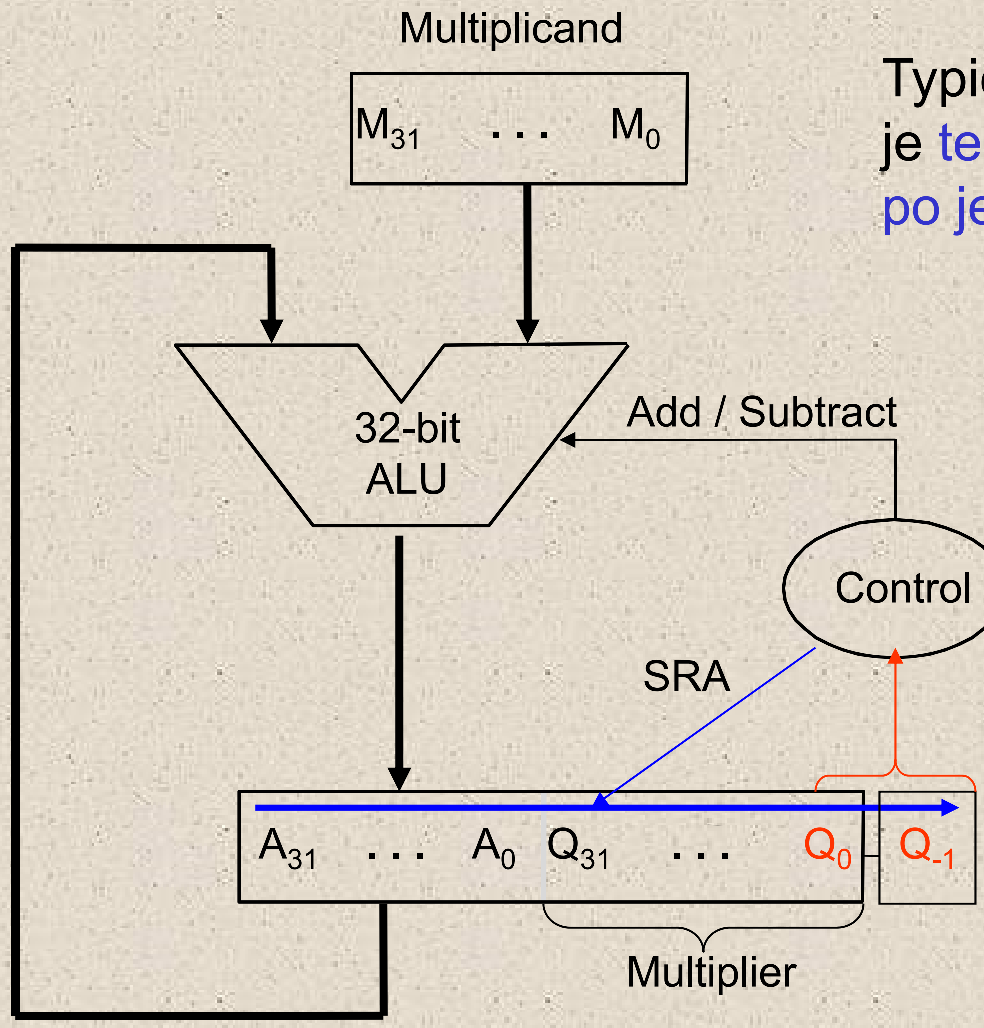
- Zpracovává kladné i záporné operandy
- Mnohem efektivnější než konvenční algoritmus

  $2^n + 2^{n-1} + 2^{n-2} + \ldots + 2^k = 2^{n+1} - 2^k$

- Je možná další optimalizace

- Operaci násobení lze implementovat pomocí hardware pro posuvy, sčítání (! odčítání)
- Instrukce násobení MIPS ignorují přetečení
  - Multu: Hi = 0
  - Mult: Hi = rozšířené znaménko z Lo

Hardware pro Boothův algoritmus



Typickým znakem Boothova algoritmu je test dvou bitů násobitele a postup po jednom bitu

Příklad

$$\begin{array}{rl} 7 & (0\ 1\ 1\ 1) \\ \times\ 3 & (0\ 0\ 1\ 1) \\ \hline \end{array}$$

	A	Q	Q_{-1}	M		
Počáteční hodnoty	0000	0011	0	0111		
	1001	0011	0	0111	A = A - M	1
	1100	1001	1	0111	Shift	
	1110	0100	1	0111	Shift	2
	0101	0100	1	0111	A = A + M	3
	0010	1010	0	0111	Shift	
	0001	0101	0	0111	Shift	4

Důkaz: kladný násobitel

- Předpokládejme *jednoduchý* kladný násobitel

  0 0 0 1 1 1 1 0 (jeden blok jedniček obklopený nulami)

  $M \times (0\,0\,0\,1\,1\,1\,1\,0) = M \times (2^4 + 2^3 + 2^2 + 2^1)$

  $= M \times (16 + 8 + 4 + 2)$

  $= M \times 30$

  5 4 3 2 1 0

- Poznámka: $\mathbf{2^n + 2^{n-1} + \ldots + 2^{n-k} = 2^{n+1} - 2^{n-k}}$

  => $M \times (0\,0\,0\,1\,1\,1\,1\,0) = M \times (2^5 - 2^1)$

- Boothův algoritmus odpovídá schématu:
  - Odečti, jestliže je nalezena kombinace (1-0)
  - Přičti, jestliže je nalezena kombinace (0-1)
- Toto pravidlo se použije na každý blok jedniček

Důkaz: záporný násobitel

- Reprezentace záporného čísla (X):

  $\{ 1\ x_{n-2}\ x_{n-3}\ .\ .\ .\ x_1\ x_0 \}$
- $X = -2^{n-1} + x_{n-2}{*}2^{n-2} + x_{n-3}{*}2^{n-3}\ \ .\ .\ .\ x_0{*}2^0$
- Předpokládejme 0 nejvíc vlevo v *k*-té pozici

  Reprezentace $X = \{ 1\ 1\ 1\ \ldots\ 10\ X_{k-1}\ \ldots\ X_0\}$

  $X = -2^{n-1} + 2^{n-2}\ .\ .\ .\ 2^{k+1} + x_{k-1}{*}2^{k-1}\ .\ .\ .\ x_0{*}2^0$

  $-2^{n-1} + 2^{n-2} + \ldots + 2^{k+1} = -2^{k+1}$

  $X = -2^{k+1} + x_{k-1}{*}2^{k-1}\ .\ .\ .\ x_0{*}2^0$

  (1-0) tato změna znamená operaci odečtení

Násobení u MIPS

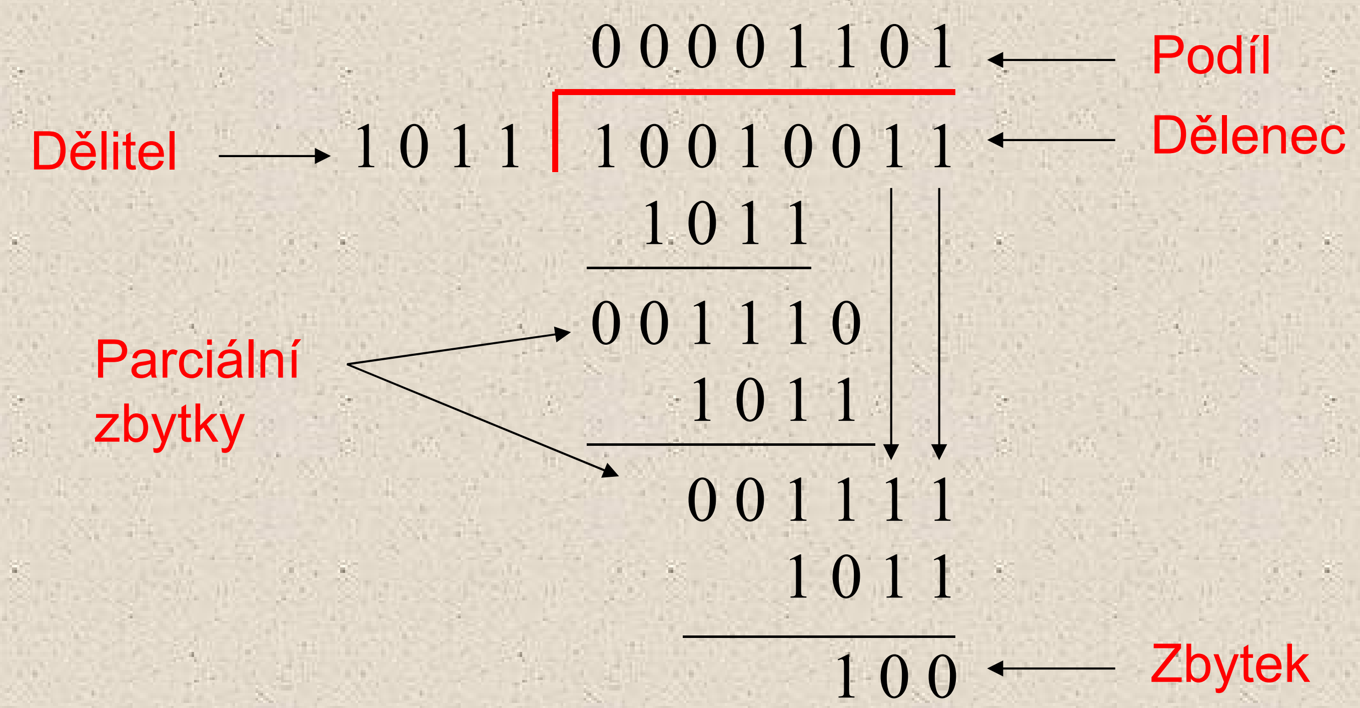
- Pro výsledek jsou vyhrazeny registry (Hi, Lo)
- Dvě instrukce pro násobení
  - Mult: se znaménkem
  - Multu: bez znaménka
- mflo, mfhi – přesune obsah Hi, Lo do obecných registrů (GPR)
- *Neprovádí se žádná hardwarová detekce přetečení*

  => Softwarová detekce přetečení

    - Hi musí být 0 pro *multu* nebo rozšířené znaménko operandu Lo pro *mult*

Dělení

- Dlouhé dělení binárních čísel integer bez znaménka



Dělenec = Podíl * Dělitel + Zbytek

Dělení

- Operace dělení je z principu sekvenční.
- Převážná většina sekvenčních metod pracuje podle rekurentního vztahu:

$$R_{j+1} = z \,.\, R_j - q_j . D$$

kde R_{j+1} ... zbytek do dalšího kroku

R_0 ... dělenec

R_j ... aktuální zbytek

q_j ... cifra podílu generovaná v j-tém kroku

z ... základ číselné soustavy

D ... dělitel

$Q = \{q_0, q_1, q_2, q_3, q_4, \dots, q_{n-2}, q_{n-1}\}$ ... podíl

Některé metody dělení

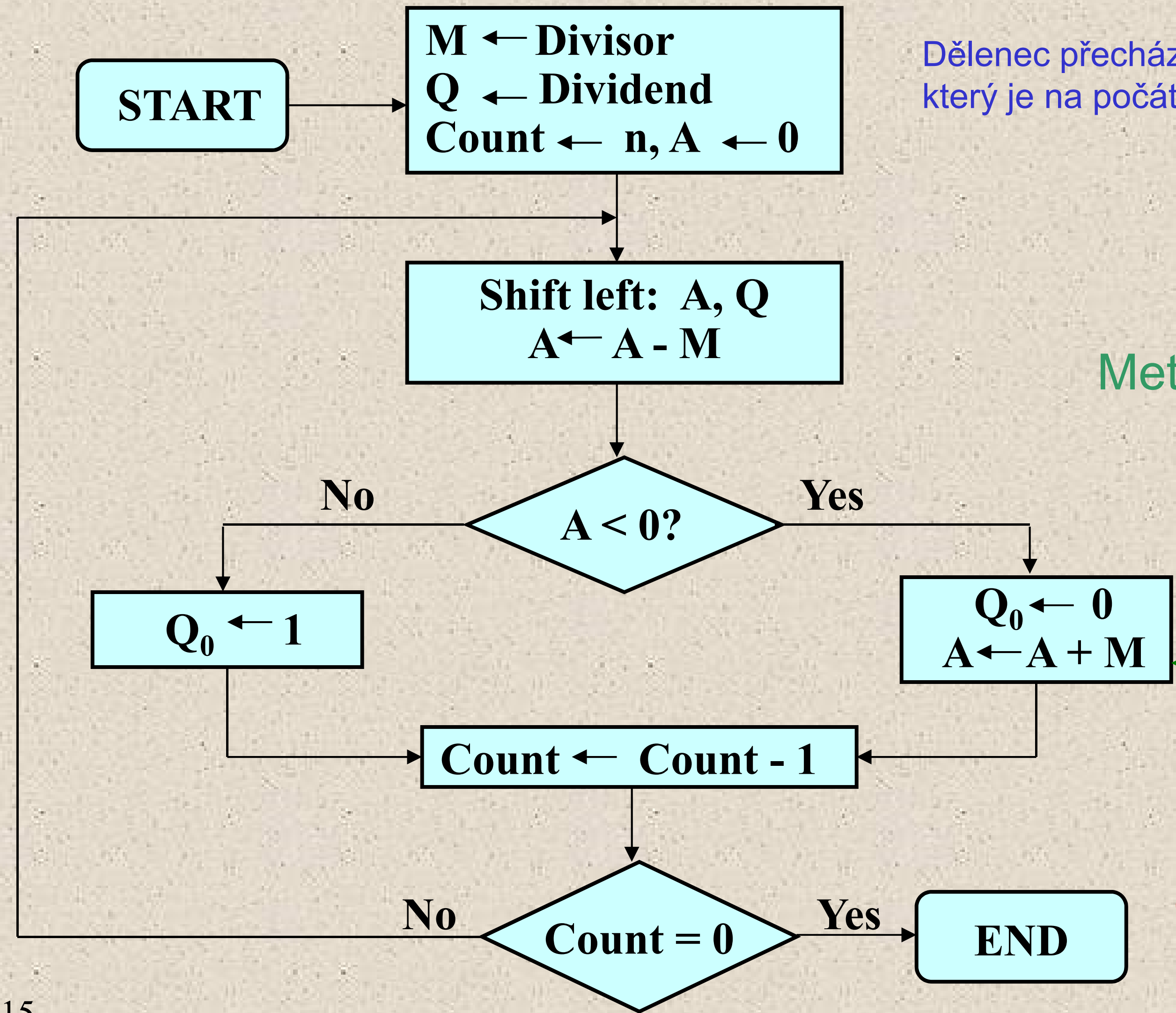
Sekvenční metody binárního dělení se hlavně odlišují množinou generovaných cifer:

$z = 2$

$q_j \in \{0, 1\}$ … metoda binárního dělení s obnovou zbytku
$q_j \in \{-1, 1\}$ … metoda binárního dělení bez obnovy zbytku
$q_j \in \{-1, 0, 1\}$ … binární metoda SRT (Sweeny-Robertson-Toucher)

Poznámka:
Metody, které generují záporné cifry výsledku, vyžadují korekční kroky. Lze je provádět většinou již v průběhu dělení (nezpůsobují další zpoždění).

Dělení bez znaménka

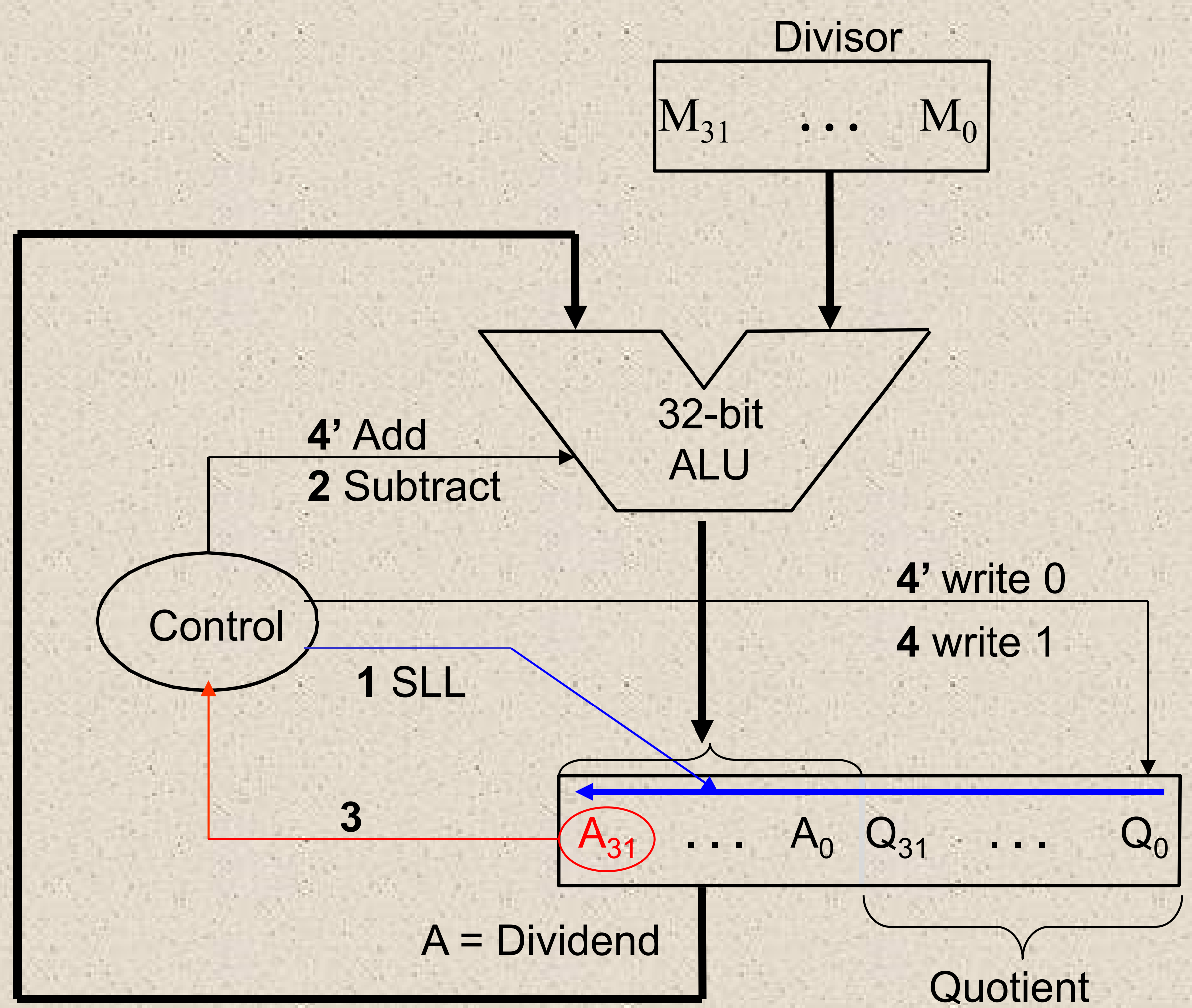


Dělenec přechází do A postupně z Q, který je na počátku volný

Metoda s obnovou zbytku

Quotient in Q
Remainder in A

Hardware pro operaci dělení



Příklady

7 / 3 :

A	Q	M = 0011	
0000	0111	Initial values	
0000	1110	Shift	1
1101	1110	A = A - M	
0000	1110	A = A + M	
0001	1100	Shift	2
1110	1100	A = A - M	
0001	1100	A = A + M	
0011	1000	Shift	3
0000	1000	A = A - M	
0000	1001	Q_0 = 1	
0001	0010	Shift	4
1110	0010	A = A - M	
0001	0010	A = A + M	

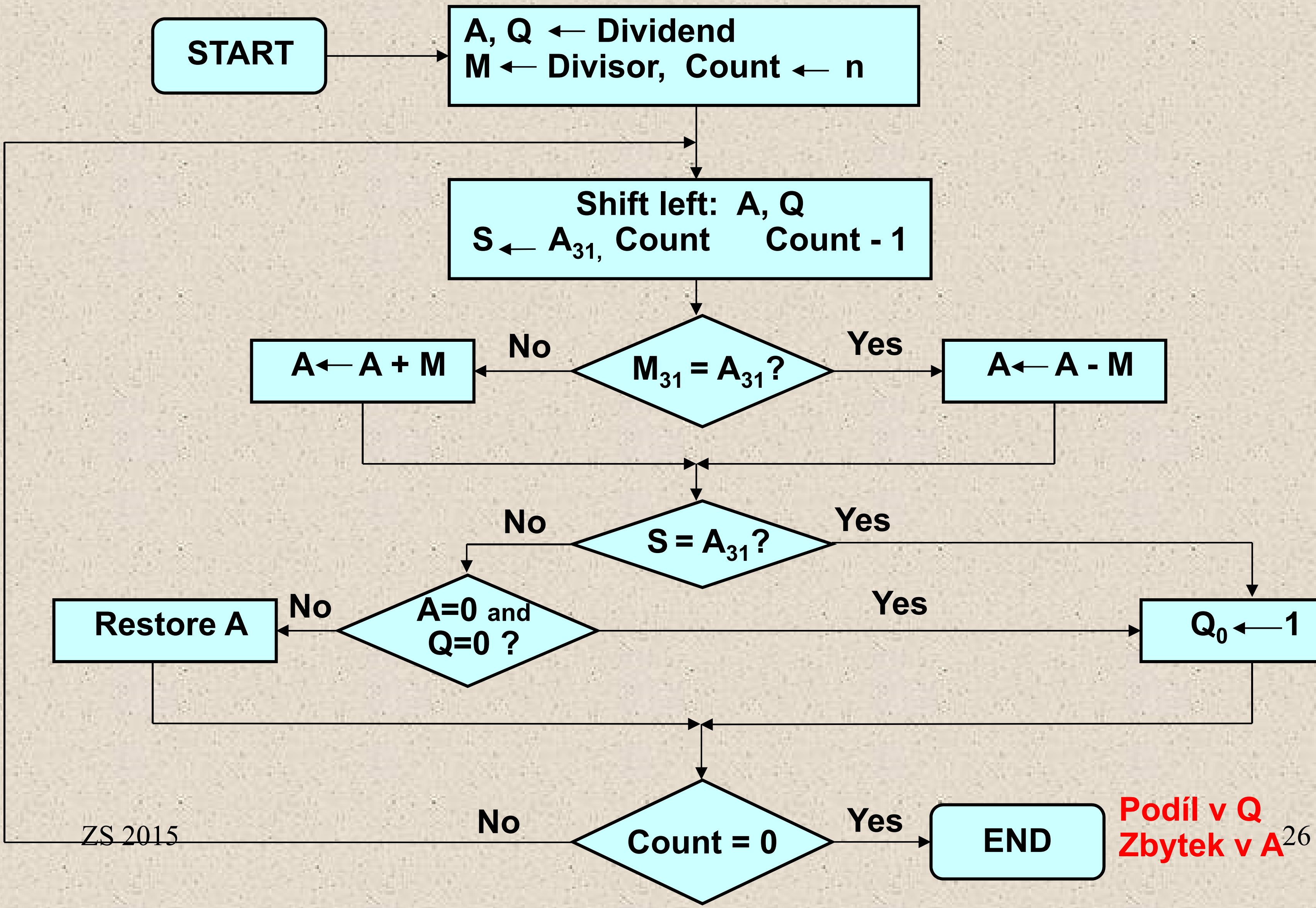
Operace dělení se znaménkem

- Jednoduché řešení
  - Negovat podíl, jestliže znaménka dělitele a dělence nejsou shodná
  - Zbytek a dělenec musí mít shodná znaménka
- Zbytek = (Dělenec – Podíl * Dělitel)

(+7) / (+3):	Q = 2;	R = 1
(-7) / (+3):	Q = -2;	R = -1
(+7) / (-3):	Q = -2;	R = 1
(-7) / (-3):	Q = 2;	R = -1

Quotient Remainder

Algoritmus dělení se znaménkem



Příklady (1/2)

A	Q	M = 0011	Step
0000	0111	Initial values	
0000	1110	Shift	1
1101		Subtract	
0000	1110	Restore	
0001	1100	Shift	2
1110		Subtract	
0001	1100	Restore	
0011	1000	Shift	3
0000		Subtract	
0000	1001	$Q_0 = 1$	
0001	0010	Shift	4
1110		Subtract	
0001	0010	Restore	

(7) / (3)

A	Q	M = 1101	Step
0000	0111	Initial values	
0000	1110	Shift	1
1101		Add	
0000	1110	Restore	
0001	1100	Shift	2
1110		Add	
0001	1100	Restore	
0011	1000	Shift	3
0000		Add	
0000	1001	$Q_0 = 1$	
0001	0010	Shift	4
1110		Add	
0001	0010	Restore	

(7) / (-3)

Příklady (2/2)

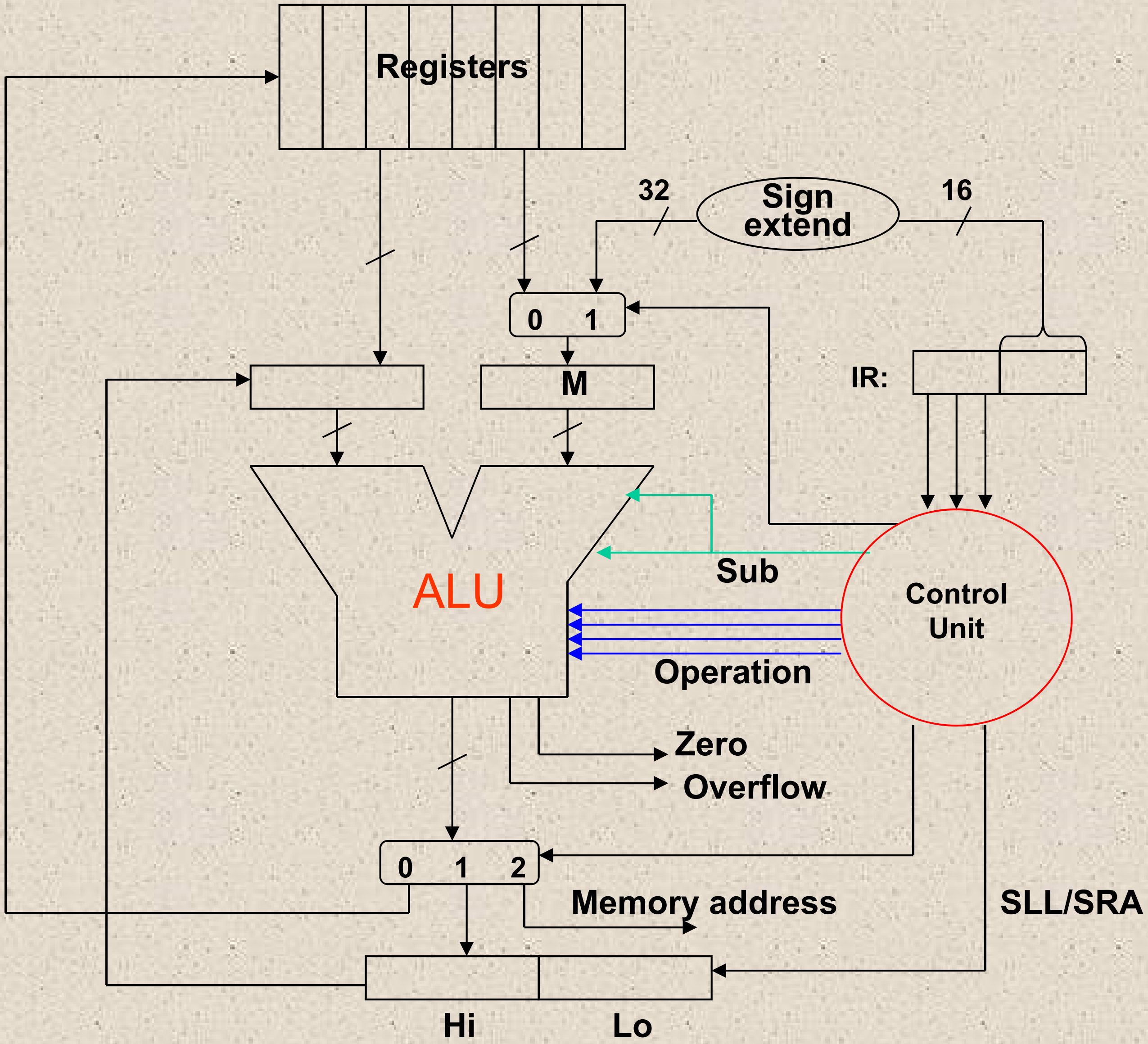
A	Q	M = 0011	
1111	1001	Initial values	
1111	0010	Shift	1
0010		Add	
1111	0010	Restore	
1110	0100	Shift	2
0001		Add	
1110	0100	Restore	
1100	1000	Shift	3
1111		Add	
1111	1001	$Q_0 = 1$	
1111	0010	Shift	4
0010		Add	
1111	0010	Restore	

(-7) / (3)

A	Q	M = 1101	
1111	1001	Initial values	
1111	0010	Shift	1
0010		Subtract	
1111	0010	Restore	
1110	0100	Shift	2
0001		Subtract	
1110	0100	Restore	
1100	1000	Shift	3
1111		Subtract	
1111	1001	$Q_0 = 1$	
1111	0010	Shift	4
0010		Subtract	
1111	0010	Restore	

(-7) / (-3)

Procesor MIPS



Závěr

- Násobení => Posuv-&-součet
- Násobení bez znaménka = násobení se znaménkem
- Pro násobení se znaménkem se používá Boothův algoritmus
- Základní verze Boothova algoritmu:
  - Test dvou bitů
  - Postup po jednom bitu
- MIPS má speciální (vyhrazené) registry (Hi, Lo) a dvě instrukce (**`mult, multu`**)

MIPS

- Operace násobení a dělení využívá existující hardware
  - ALU a posuvovou jednotku
- Extra hardware: 64-bitový registr pro operace SLL/SRA
  - Hi obsahuje zbytek (mfhi)
  - Lo obsahuje podíl (mflo)
- Instrukce
  - Div: dělení se znaménkem
  - Divu: dělení bez znaménka
- MIPS ignoruje přetečení ?
- Dělení nulou se musí testovat softwarově !