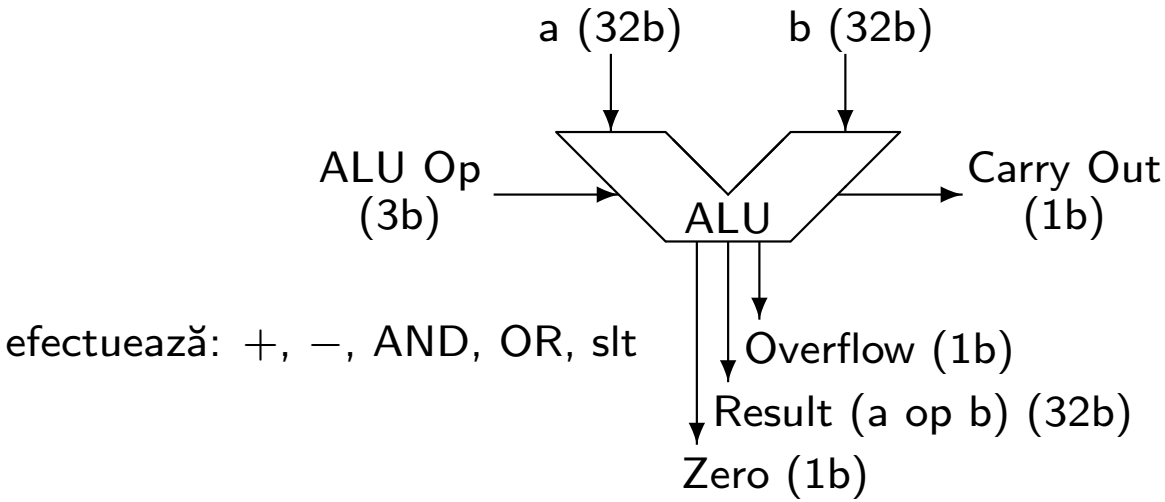
Unitatea aritmetico-logică (ALU)



Unitatea aritmetico-logică (ALU)

ALU efectuează operații într-un ciclu.
În ALU sunt implementate câteva operații simple ($+$, $-$, AND, OR, slt), ceea ce permite efectuarea de cicluri scurte.

Înmulțirea și împărțirea sunt operații complexe.
Ele ar putea fi implementate într-un ciclu, în ALU, dar atunci structura acestuia s-ar dezvolta foarte mult pe verticală, ceea ce ar duce la lungirea ciclului.

De aceea, înmulțirea și împărțirea sunt implementate într-o unitate separată, folosind mai multe cicluri ALU, conform unor anumiți algoritmi.

Prezentăm în continuare câțiva algoritmi de înmulțire și împărțire.

Înmulțire - metoda 1

Reproduce metoda clasică, aplicată manual:

De ex. pt. a calcula $\overbrace{100}^{D} \times \overbrace{101}^{I} = \overbrace{11110}^{P}$ având dim. word-ului $n = 4$, efectuăm:

$$\begin{array}{r} 110\times \\ 101 \\ \hline 110 \\ 110 \\ \hline 11110 \end{array}$$

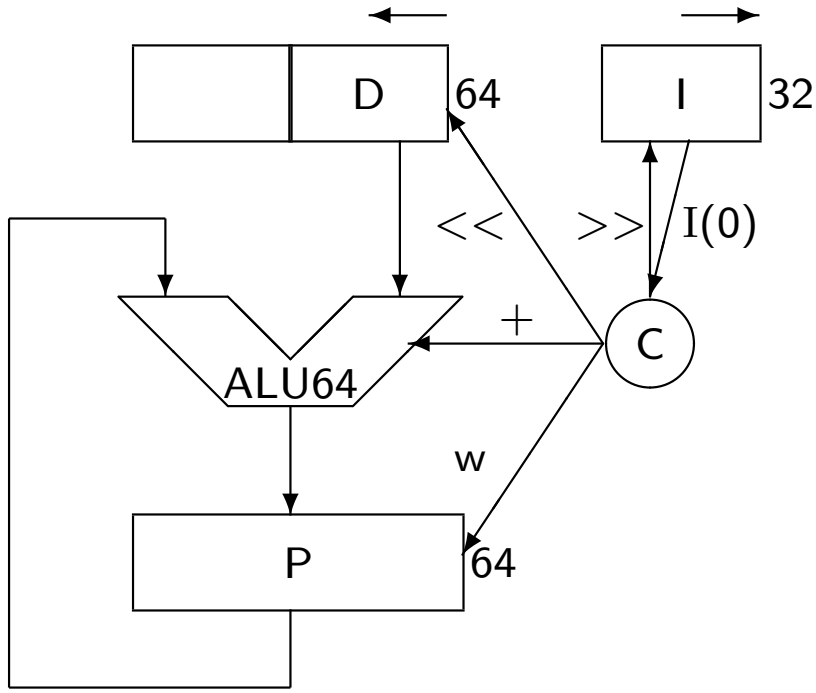
← Parcurgem I dr. $\rightarrow$ stg. și pentru fiecare 1 copiez D shiftat; apoi adun.

← P are dim. max. $2n$.

Reformulare cu acțiuni mai apropiate de operațiile mașină:
Shiftez de n ori I la dreapta și D la stânga;
de fiecare dată când ultimul bit din I (adică I(0)) este 1, adun D la P (în poziția curentă).

Înmulțire - metoda 1

$D_{64},\ I_{32},\ P_{64} := 0$
repetă de 32 $\times$:
  dacă $I(0) = 1$
    $P := P + D$
  $D := D << 1$
  $I := I >> 1$



Aplicație: Calculați $6 \times 5 = 30$, $n = 4$. Completați tabelul următor, fiecare coloană dublă conține valorile regiștrilor LA SFÂRȘITUL unei etape:

		Inițial		Iterația 1		Iterația 2		Iterația 3		Iterația 4	
>>	I	-	0101	-		-		-		-	
<<	D	0000	0110								
+	P	0000	0000								

Înmulțire - metoda 1

Analiza algoritmului:

Se efectuează 32 pași $\times$ 3 operații $\approx$ 100 cicluri pe instrucțiune.

Analizele statistice arată că $+$, $-$ sunt de 5 - 100 $\times$ mai frecvente decât $\star$.
Atunci, cu regula "execută rapid operațiile frecvente", rezultă că această implementare pentru înmulțire este acceptabilă.

Totuși, dorim și putem obține implementări mai eficiente.

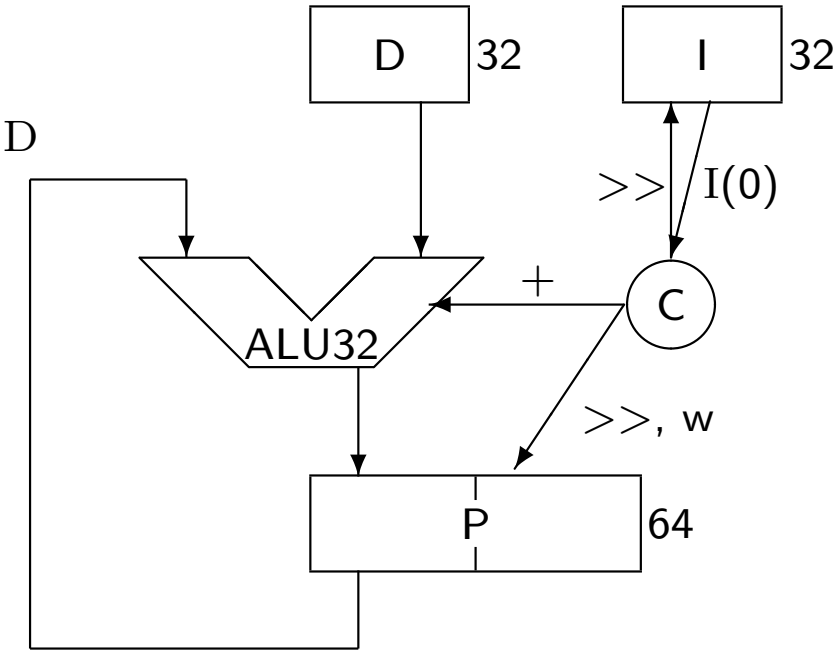
Înmulțire - metoda 2

Obs. că prima jumătate a lui D este nefolosită (e necesară ca să putem aduna pe 64b).

Soluția: facem D de 32b și în loc să deplasăm D deasupra lui P, deplasăm P pe sub D; întotdeauna vom aduna D cu prima jumătate a lui P (adunare pe 32b).

Înmulțire - metoda 2

$D_{32},\ I_{32},\ P_{64} := 0$
repetă de 32 $\times$:
  dacă $I(0) = 1$
    $P[63-32] := P[63-32] + D$
  $P := P >> 1$
  $I := I >> 1$



Aplicație: Calculați $6 \times 5 = 30$, $n = 4$. Completați tabelul următor, fiecare coloană dublă conține valorile regiștrilor LA SFÂRȘITUL unei etape:

		Inițial		Iterația 1		Iterația 2		Iterația 3		Iterația 4	
>>	I	-	0101	-		-		-		-	
	D	0110	-		-		-		-		-
+	P	0000	0000								
>>											

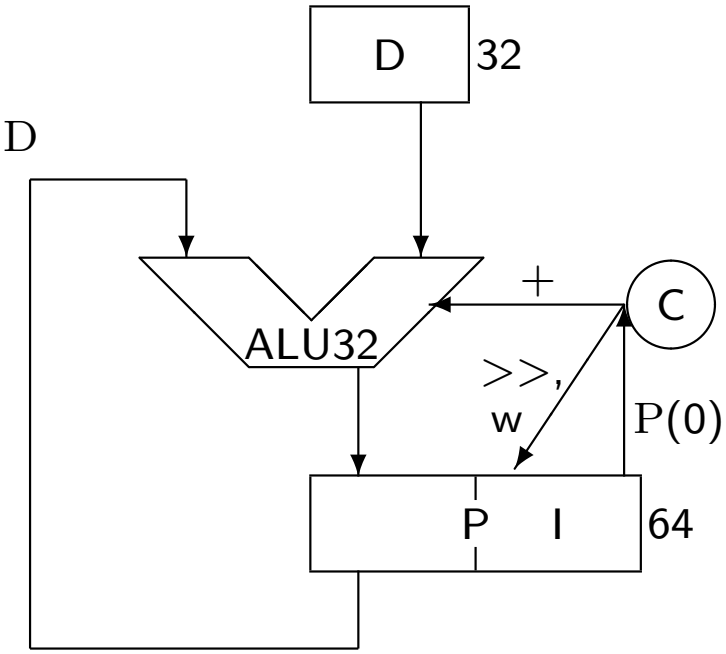
Înmulțire - metoda 3 (finală)

Obs. că jumătatea inferioară (LO) a lui P se consumă spre dreapta în același ritm ca I.

Atunci putem pune I în jumătatea inferioară a lui P; în rest este la fel.

Înmulțire - metoda 3 (finală)

$D_{32},\ P_{64} := [0, \ldots, 0, I_{32}]$
repetă de 32 $\times$:
  dacă $P(0) = 1$
    $P[63-32] := P[63-32] + D$
  $P := P >> 1$



Aplicație: Calculați $6 \times 5 = 30$, $n = 4$. Completați tabelul următor, fiecare coloană dublă conține valorile regiștrilor LA SFÂRȘITUL unei etape:

		Inițial		Iterația 1		Iterația 2		Iterația 3		Iterația 4	
	D	0110	-		-		-		-		-
+, >>	P,I	0000	0101								

Înmulțirea cu semn

Înmulțirea cu semn este asemănătoare, cu următoarele diferențe:

- se lucrează pe 31 biți, deci se fac 31 iterații, neglijând bitul de semn;
- în final, semnul produsului este XOR între biții de semn ai factorilor (deci este 1 dacă au semne diferite ("-") și 0 dacă au același semn ("+"));
- algoritmul 3 (final) funcționează corect și pentru numere cu semn, dar la shiftare trebuie extins semnul produsului (shiftare aritmetică).

Înmulțire - algoritmul Booth

Face câteva îmbunătățiri plecând de la următoarele observații:
- înmulțirea cu un grup de 0 din I se reduce la shiftări;
- înmulțirea cu un grup de 1 din I: $\underbrace{1\ldots1}_{k} = 2^k - 1 = 1\underbrace{0\ldots0}_{k} - 0\ldots01$ se reduce la adunarea lui D shiftat cu k pentru bitul 1 din grupul $10\ldots0$ și o scădere a lui D pentru bitul 1 din grupul $0\ldots01$.

$$\text{De ex. } \overset{\text{D}}{0010} \star \underbrace{\overset{\text{I}}{0110}}_{} = 0010 \star 1000 - 0010 \star 0010$$
$$= 1000 - 0010$$

Practic shiftăm I la dreapta cu câte 1 și luăm decizii în funcție de cu încep/continuă/se termină grupurile de 1 sau 0 (algoritmul și circuitul seamănă cu cel de la metoda 3):

01100 → se adaugă un bit fictiv 0, pentru a avea un context;

01100 ⇒ se shiftează P;
00110 ⇒ se scade D (începe un grup de 1 și se face întâi scăderea lui D);
00011 ⇒ se shiftează P (continuă grupul de 1);
00001 ⇒ se adună D (se termină grupul de 1 și se face adunarea lui D);
00000 ⇒ se shiftează P.

Înmulțire - algoritmul Booth

D_{32}, $P[63,\ldots,-1] := [\overbrace{0,\ldots,0}^{32}, I_{32}, 0]$

repetă de 32 ×:

- testează (P(0),P(−1))
  - cazul 01: $P[63-32] := P[63-32] + D$
  - cazul 10: $P[63-32] := P[63-32] - D$
- $P := P >> 1$ (shift aritmetic)

Algoritmul și circuitul le adaptează pe cele de la metoda 3.

Algoritmul funcționează corect și pentru numere negative și e performant (se poate folosi pe grupuri de biți pentru a construi înmulțitoare rapide.)

Aplicație: Calculați $5 \times 6 = 30$ ($\underset{D}{101} \times \underset{I}{110} = \underset{P}{11110}$), $n = 4$.

Completați tabelul următor, fiecare linie conține valorile regiștrilor LA SFÂRȘITUL unei etape:

Inițial	0000 0101	0110	0
Iterația 1			
Iterația 2			
Iterația 3			
Iterația 4			

Împărțire - metoda 1

Trebuie efectuat D:I $\rightarrow$ C,R $\quad (D = I \times C + R, \;\; R < I)$.

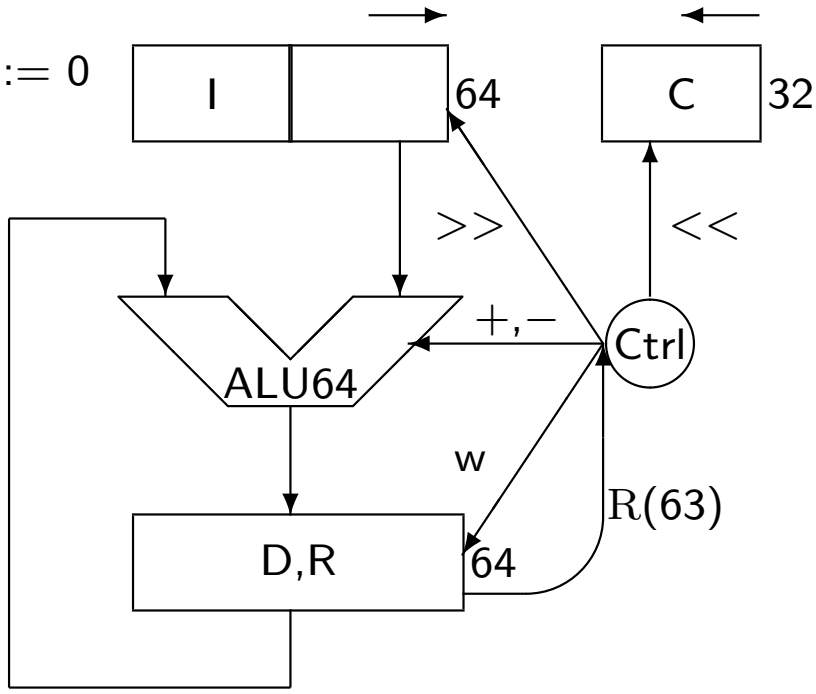
Metoda 1 reproduce metoda clasică, aplicată manual, de exemplu:

$$\begin{array}{rl|l} D = & 1001010 & 1000 \;\; = I \\ \cline{2-3} -1000 & & 1001 \;\; = C \\ \cline{1-1} \quad 10 & & \\ \quad 101 & & \\ \quad 1010 & & \\ -\;1000 & & \\ \cline{1-1} \quad\;\; 10 & = R & \end{array}$$

D.p.v al mașinii:
– a vedea dacă I "se cuprinde" revine la a scădea D − I și a compara cu 0 (i.e. a testa bitul cel mai semnificativ);
dacă dă ≥ 0, adăugăm 1 la C;
dacă dă < 0, adunăm D + I la loc și adăugăm 0 la C;
– adăugarea unei cifre la C înseamnă $<< 1 +$ cifră;
– coborârea cifrei următoare înseamnă shiftarea lui I pe sub D la dreapta (metoda 1) sau a lui D pe deasupra lui I la stânga (metodele 2,3), pentru a face altă suprapunere la scădere.

Împărțire - metoda 1

R_{64}:=D, I_{64}:=[I, $\overbrace{0,\ldots,0}^{32}$], $C_{32} := 0$
repetă de 33 ×:
  $R := R - I$
  dacă R $\geq$ 0 (i.e. $R(63) = 0$)
    $C := C << 1 + 1$
  altfel
    $R := R + I$
    $C := C << 1 + 0$
  □
  $I := I >> 1$
□



Aplicație: Calculați 7:3 (adică 111:11), $n = 4$. Completați tabelul următor, fiecare coloană dublă conține valorile regiștrilor LA SFÂRȘITUL unei etape:

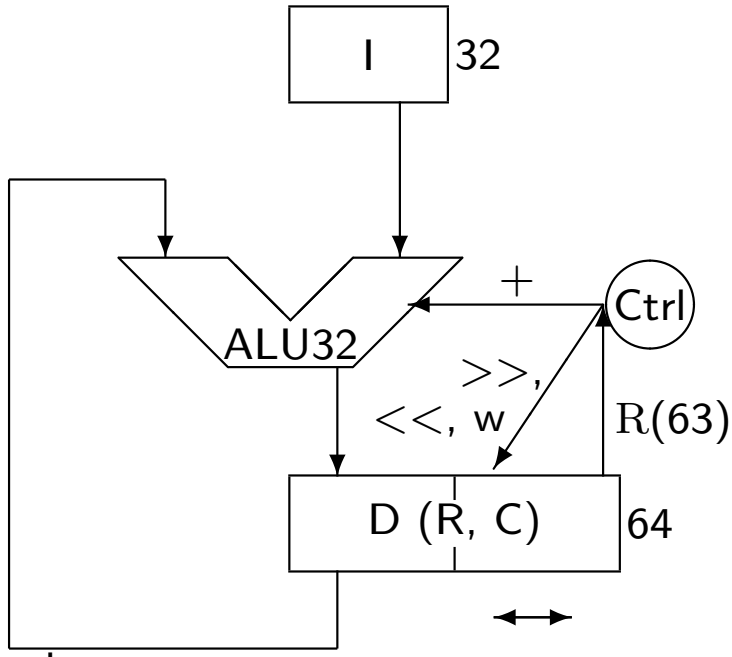
		Inițial		Iterația 1		Iterația 2		Iterația 3		Iterația 4		Iterația 5	
<<	C	-	0000	-		-		-		-		-	
	R,D	0000	0111										
>>	I	0011	0000										

Împărțire - metoda 1

Obs. că doar jumătate din I conține informație utilă; atunci, putem folosi un ALU32 și să shiftăm R pe deasupra lui I la stânga.
Apoi obs. că algoritmul nu poate produce un 1 în prima fază, căci rezultatul ar fi prea lung pentru C și n-ar încăpea într-un registru de 32b (avem $n+1$ iterații, deci am avea un cât de forma $\underbrace{1\ldots}_{33 \text{ cifre}}$ și nu ar încăpea).
Soluția: se permută operațiile de shiftare și scădere (se face întâi shiftarea și apoi scăderea), eliminându-se o iterație; la sfârșit restul este în jumătatea stângă a lui R.
Se obține **metoda 2**, dar nu o prezentăm, ci trecem direct la **metoda 3**, unde în plus se pune C în jumătatea dreaptă alui R - cum R și C se shiftează sincron cu 1 la stânga, nu se pierde nimic din R, C.
Singura problemă este că la sfârșit jumătatea stângă a lui R este prea shiftată și se shiftează la dreapta cu 1.

Împărțire - metoda 3

R_{64}:=D, I_{32}
$R := R << 1$
repetă de 32 $\times$:
  $R[63-32] := R[63-32] - I$
  dacă R ≥ 0 (i.e. $R(63) = 0$)
    $R := R << 1 + 1$
  altfel
    $R[63-32] := R[63-32] + I$
    $R := R << 1 + 0$
  □
□
  $R[63-32] := R[63-32] >> 1$



$\Downarrow$

$R[63-32]$ = restul, $R[31-0]$ = câtul

Aplicație: Calculați 7:3 (adică 111:11), $n = 4$, completând tabelul următor:

Inițial	0000 0011	0111
$R << 1$		
Iterația 1		
Iterația 2		

$R[63-32] >> 1$

Iterația 3		
Iterația 4		