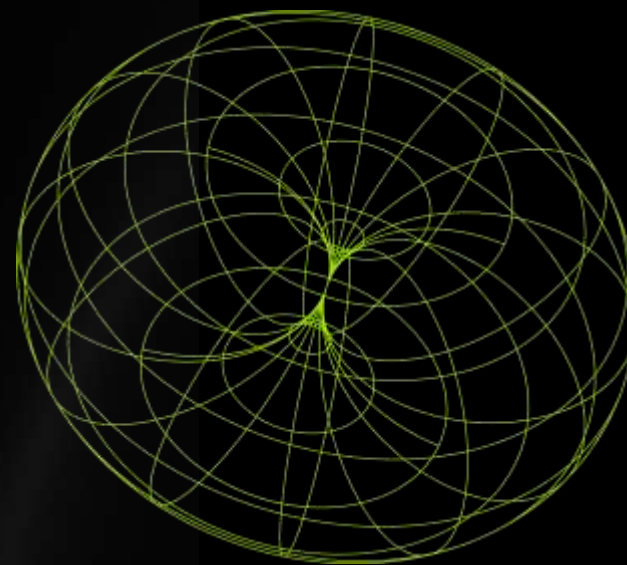
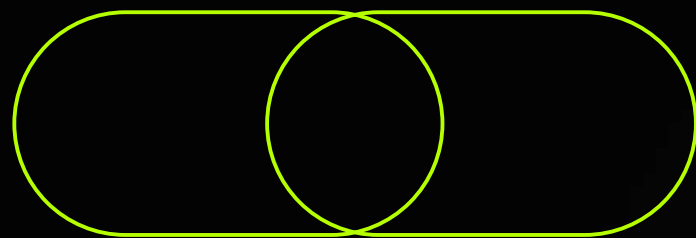
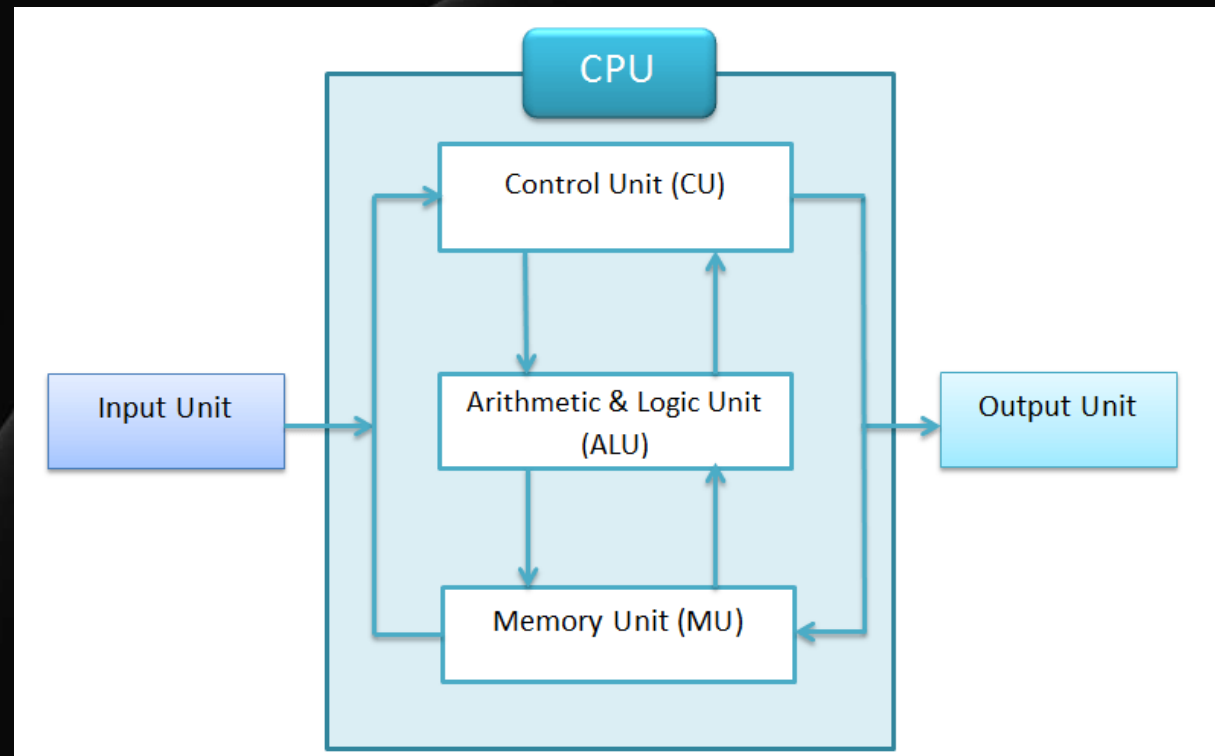


ARSITEKTUR DAN ORGANISASI KOMPUTER

A R I T M A T I K A K O M P U T E R



PENDAHULUAN



ALU, singkatan dari Arithmetic And Logic Unit (bahasa Indonesia: unit aritmatika dan logika), adalah salah satu bagian dalam dari sebuah mikroprosesor yang berfungsi untuk melakukan operasi hitungan aritmatika dan logika. Contoh operasi aritmatika adalah operasi penjumlahan dan pengurangan, sedangkan contoh operasi logika adalah logika AND dan OR. tugas utama dari ALU (Arithmetic And Logic Unit) adalah melakukan semua perhitungan aritmatika atau matematika yang terjadi sesuai dengan instruksi program. ALU melakukan operasi aritmatika yang lainnya. Seperti pengurangan, pengurangan, dan pembagian dilakukan dengan dasar penjumlahan.

BILANGAN

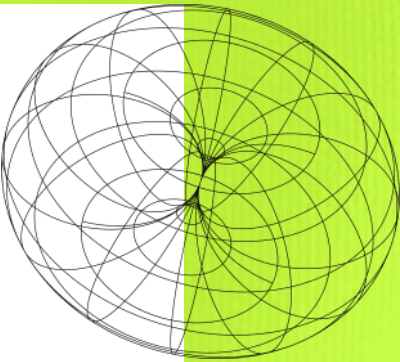
Karena transaksi elektronik logika dengan arus yang sedang aktif atau tidak aktif telah ditemukan, maka untuk mewakili kuantitas dapat dengan mudah dimengerti oleh aritmatika komputer ketika nilainya diubah dalam bentuk biner. Jadi, daripada harus berbeda sepuluh angka, 0, 1, 2, 3, 4, 5, 6, 7, 8, dan 9, di aritmetika biner, hanya ada dua digit berbeda, 0 dan 1. Ketika pindah ke kolom berikutnya, bukan mewakili angka kuantitas yang sepuluh kali lebih besar, hanya merupakan sebuah kuantitas yang dua kali lebih besar. Dengan demikian, angka pertama ditulis dalam biner sebagai berikut:

Desimal		Biner
Nol	0	0
Satu	1	1
Dua	2	10
Tiga	3	11
Empat	4	100
Lima	5	101
Enam	6	110
Tujuh	7	111
Delapan	8	1000
Sembilan	9	1001
Sepuluh	10	1010
Sebelas	11	1011
Dua belas	12	1100

BILANGAN POSITIF

Seandainya semua integer positif, konversi ke biner biasa tinggal disesuaikan dengan panjang bit register yang tersedia. Misal data akan disimpan ke dalam register 8-bit :

00000000	:	0
00000001	:	1
00101001	:	41
10000000	:	128
11111111	:	255

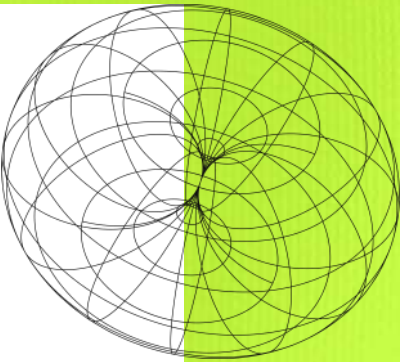


BILANGAN NEGATIF

Membahas bilangan negatif, ternyata ada masalah dalam konversi bilangan biner negatif. Lihat saja contoh di bawah ini :

+18	:	00010010
-18	:	10010010
+10	:	00001010
-10	:	10001010
+0	:	00000000
-0	:	10000000

Dari contoh +18 dan -18 masih tidak ada masalah, begitu juga dengan contoh pada +10 dan -10. Lalu bagaimana dengan +0 dan -0? Padahal hanya ada 1 nilai 0, tidak ada nol negatif dan nol positif. Maka dari itu ada solusi dalam mengatasi kekurangan pada konversi bilangan negatif ini. Solusinya adalah dengan merepresentasikan bilangan biner dengan menggunakan komplement-2.



1. Tentukan biner positif dari 18.
2. Negasikan biner dari 18.
3. Jumlahkan hasil negasi dari biner 18 dengan 1

```
1. Biner dari      18 : 00010010
2. Negasi biner dari 18 : 11101101
3. Jumlahkan hasil negasi : _____ 1+

Hasil penjumlahan negasi : 11101110

Jadi, biner dari -18 adalah 11101110
```

Sebagai informasi, metode ini digunakan di komputer sekarang. Metode konversi bilangan biner menggunakan komplemen 2 hanya memiliki 1 biner yang bernilai desimal 0 sehingga lebih unggul dibanding metode sebelumnya. Tetapi, ada kekurangan yang dihasilkan dari metode ini. Terjadi ketimpangan representasi nilai negatif dan positif untuk jumlah bit tertentu. Misal untuk 8-bit, range bilangan bulat yang terwakili adalah -128 hingga 127 (bukan 128, tapi 128-1)

Range bilangan : $(-n)$ sampai $(n-1)$

Inilah yang terjadi di komputer kita. Silahkan cek range bilangan untuk setiap tipe data yang dimengerti oleh komputer.

KONVERSI ANTARA PANJANG BIT YANG BERBEDA

Mungkin saja terdapat perbedaan jumlah bit antar register. Ada yang berjumlah 8 bit hingga 16 bit. Ini adalah peraturan konversi antara panjang bit yang berbeda : Pada bilangan positif harus ditambahkan dengan nilai 0 (digaris bawahi) di bagian depan. Contoh :

```
+18 :      00010010
+18 : 00000000 00010010
```

Sedangkan pada bilangan negatif harus ditambahkan dengan nilai 1 (digaris bawahi) di bagian depan. Contoh :

```
-18 :      10010010
-18 : 11111111 10010010
```



OPERASI ARITMATIKA

DALAM OPERASI PENJUMLAHAN DAN PENGURANGAN, KOMPUTER HANYA MENGGUNAKAN RANGKAIAN PENJUMLAHAN KARENA UNTUK OPERASI PENGURANGAN BISA DISELESAIKAN DENGAN SOLUSI :

$$8 - 2 = \underline{8 + (-2)}$$

PADA CONTOH DI ATAS, KOMPUTER AKAN MENCARI NILAI MINUS 2 KEMUDIAN DITAMBAHKAN DENGAN 8. SEHINGGA NILAI YANG DIHASILKAN AKAN SAMA DENGAN 8 DIKURANGI 2.

PENJUMLAHAN



SEPERTI OPERASI PENJUMLAHAN PADA BILANGAN DESIMAL, KITA BISA MENJUMLAHKAN BEBERAPA BILANGAN BINER DENGAN MUDAH. SIMAK CONTOH BERIKUT :

```
8 : 1000
2 : 0010 +
-----
10: 1010
```

PADA CONTOH DI SAMPING TIDAK ADA ATURAN KHUSUS KARENA HANYA ADA PENJUMLAHAN ANTARA 0 DENGAN 0 DAN ANTARA 0 DENGAN 1. LALU BAGAIMANA UNTUK PENJUMLAHAN BIT 1 DENGAN 1? SIMAK CONTOH BERIKUT :

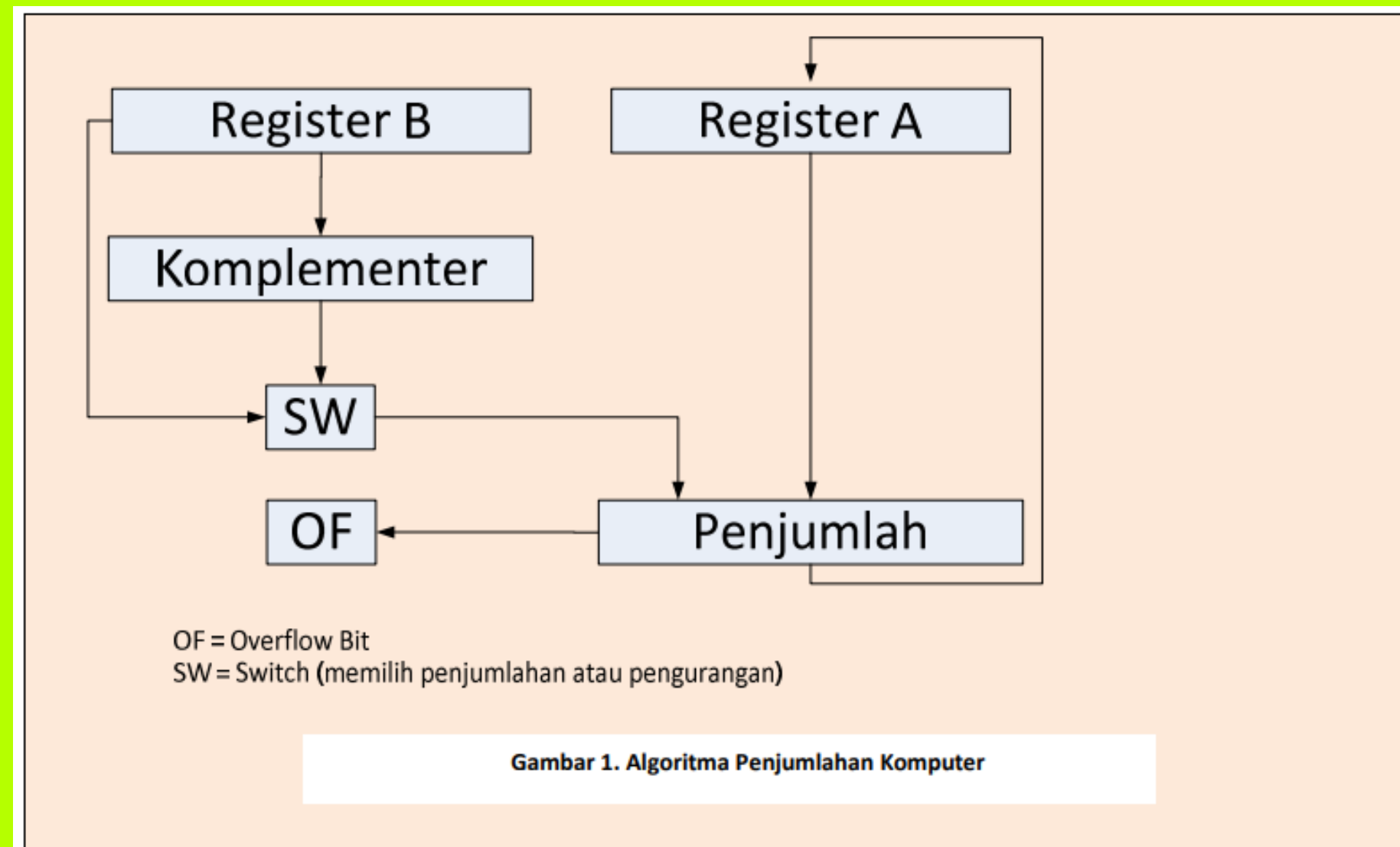
TERDAPAT KEKHUSUSAN DALAM PENJUMLAHAN BIT 1 DENGAN 1. HASIL DARI PENJUMLAHAN BIT 1 DENGAN 1 ADALAH NOL. KEMUDIAN BINER DI SEBELAH KIRINYA AKAN DITAMBAH DENGAN BIT BERNILAI 1 YANG MERUPAKAN SIMPANAN DARI BIT YANG DIJUMLAHKAN TERSEBUT.

```
10: 1010
2 : 0010 +
-----
12: 1100
```

PENGURANGAN

PENGURANGAN. YANG DIKENAL OLEH ARITMATIKA KOMPUTER ADALAH PENJUMLAHAN DARI BILANGAN NEGATIF. LANGKAH UNTUK PENGURANGAN DI ARITMATIKA KOMPUTER ADALAH :

- 1.MENCARI NILAI BINER NEGATIF DARI BILANGAN PENGURANG.
- 2.MENJUMLAHKAN BILANGAN YANG DIKURANGI DENGAN BILANGAN NEGATIF DARI PENGURANG.



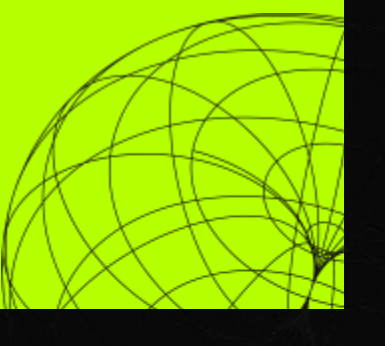


PERKALIAN

OPERASI PERKALIAN LEBIH RUMIT DIBANDING OPERASI PENJUMLAHAN ATAU PENGURANGAN, BAIK DALAM HARDWARE MAUPUN SOFTWARE. ADA BEBERAPA JENIS ALGORITMA YANG DIGUNAKAN UNTUK OPERASI PERKALIAN DALAM KOMPUTER :

- 1.PERKALIAN UNSIGNED INTEGER**
- 2.PERKALIAN KOMPLEMEN-2**
- 3.PERKALIAN MENGGUNAKAN ALGORITMA BOOTH**

APAPUN ALGORITMA YANG DIPAKAI, ADA ISTILAH UNIVERSAL YANG DIGUNAKAN UNTUK OPERASI PERKALIAN DALAM KOMPUTER. CONTOHNYA UNTUK PERKALIAN ANTARA 2 DAN 3 YANG MENGHASILKAN 6. ANGKA 2 MERUPAKAN MULTIPLICAND, ANGKA 3 MERUPAKAN MULTIPLIER, DAN ANGKA 6 MERUPAKAN PRODUCT DARI PERKALIAN TERSEBUT.



PERKALIAN UNSIGNED INTEGER

Perkalian menggunakan algoritma unsigned integer hampir sama dengan operasi perkalian untuk bilangan desimal. Bedanya, dalam operasi ini digunakan bilangan biner dengan aturan AND. Contoh, perkalian antara 11 dan 13 akan diproses seperti berikut :

```

    1011 (11)
    1101 (13) x
    -----
    1011
   0000
  1011
 1011
+-----+
10001111 (143)
```

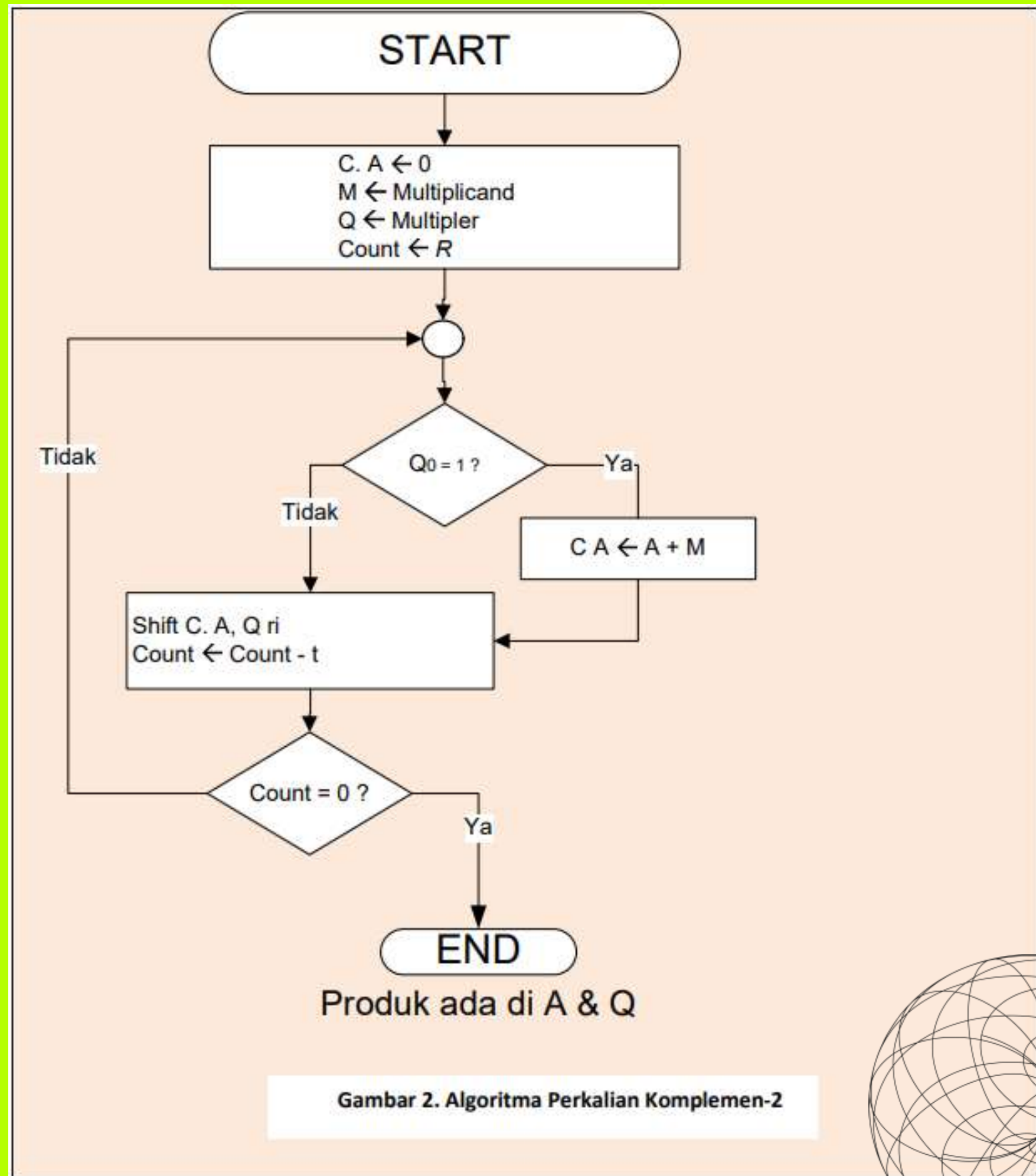
Pengalian meliputi pembentukan beberapa perkalian parsial untuk setiap digit dalam multiplier. Perkalian parsial ini kemudian dijumlahkan untuk mendapatkan hasil pengalian akhir. Pengalian dua buah integer biner n-bit menghasilkan product sampai 2n-bit.

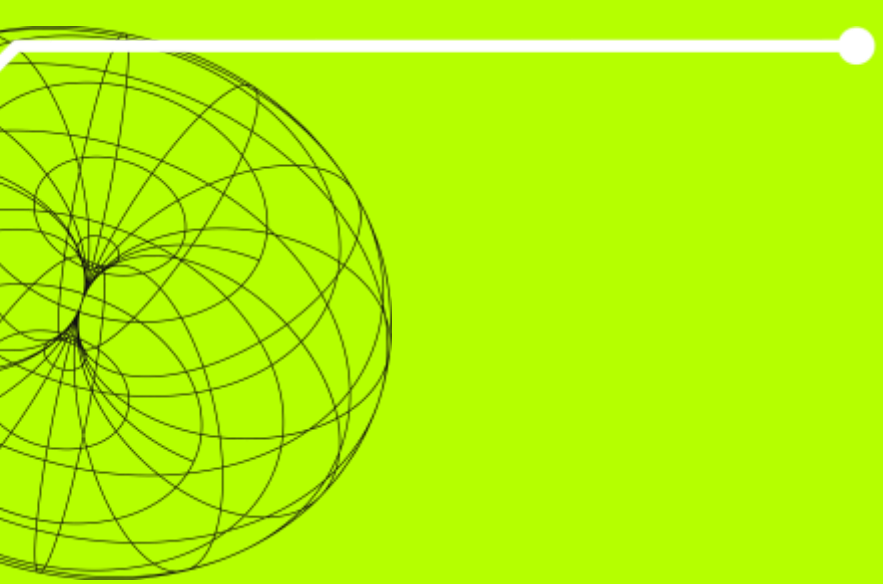
PERKALIAN KOMPLEMEN-2

Perkalian komplement-2 sedikit lebih rumit dibandingkan dengan algoritma sebelumnya. Berikut adalah algoritma untuk perkalian yang menggunakan komplement-2 :

1. Control Logic membaca bit-bit multiplier satu persatu.
2. Bila $Q_0 = 1$, multiplicand ditambahkan ke register A; hasilnya disimpan ke register A; setelah itu seluruh bit di register C, A, dan Q digeser ke kanan 1 bit.
3. Bila $Q_0 = 0$, tidak terjadi penambahan; seluruh bit di register C, A, dan Q digeser ke kanan 1 bit.
4. Proses tersebut dilakukan secara berulang untuk setiap bit multiplier.
5. Hasil perkalian akhir tersimpan di register A dan Q.

Secara singkat,
algoritmanya bisa
digambarkan sebagai
berikut





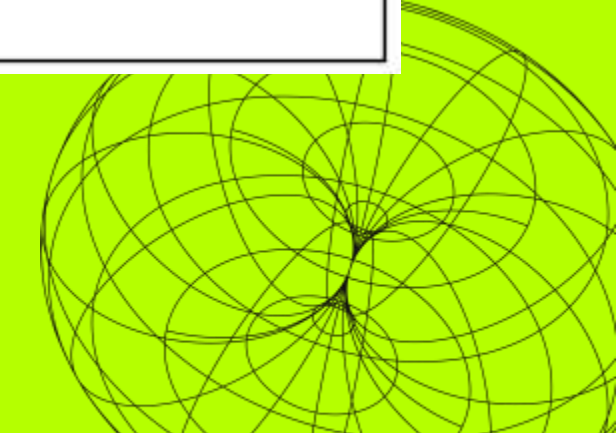
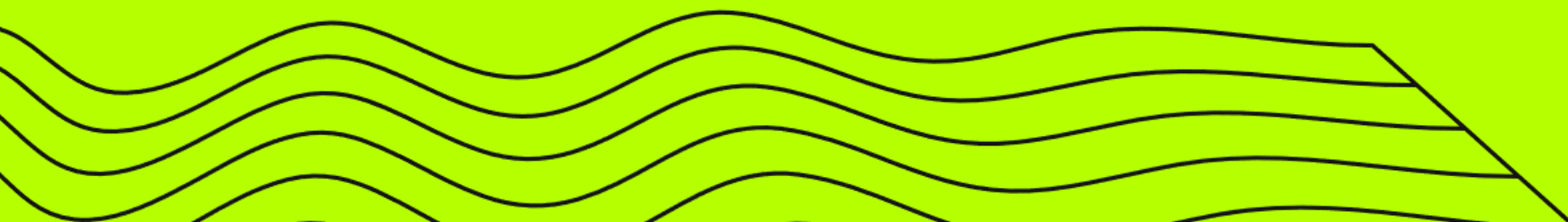
**Masuk ke implementasi,
berikut adalah contoh dari
operasi perkalian antara 11
dan 13 menggunakan
algoritma komplemen-2 :**

11 : 1011 → Multiplicand (Adder)

13 : 1101 → Multiplier (Q)

C	A	Q		
0	0000	110 <u>1</u>	Initial Value	
0	1011	1101	Hasil penjumlahan	} Siklus I
0	0101	111 <u>0</u>	Hasil geser kanan	
0	0010	111 <u>1</u>	Hasil geser kanan	Siklus II
0	1101	1111	Hasil penjumlahan	} Siklus III
0	0110	111 <u>1</u>	Hasil geser kanan	
1	0001	1111	Hasil penjumlahan	} Siklus IV
0	<u>1000</u>	<u>1111</u>	Hasil geser kanan	

Product perkalian antara 1011 dan 1101 adalah 1000 1111

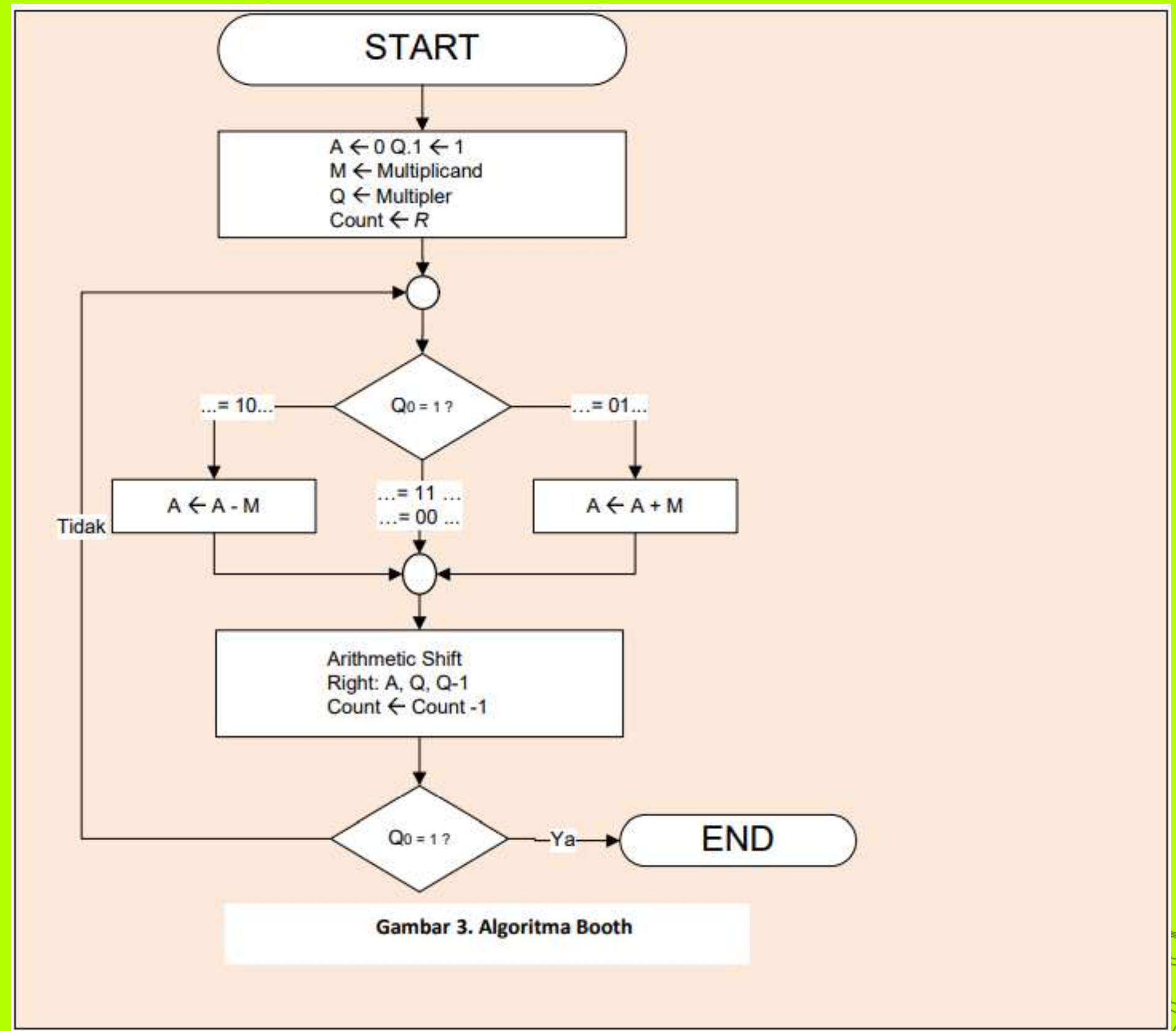


ALGORITMA BOOTH

Ada algoritma lain untuk operasi perkalian, yakni algoritma Booth. Di algoritma Booth terdapat register Q (multiplier), M (multiplicand), A (accumulator), dan register 1-bit di kanan yang ditandai dengan Q1. Hasil perkalian disimpan di register A dan Q. Berikut adalah algoritma Booth yang digunakan dalam menyelesaikan operasi perkalian :

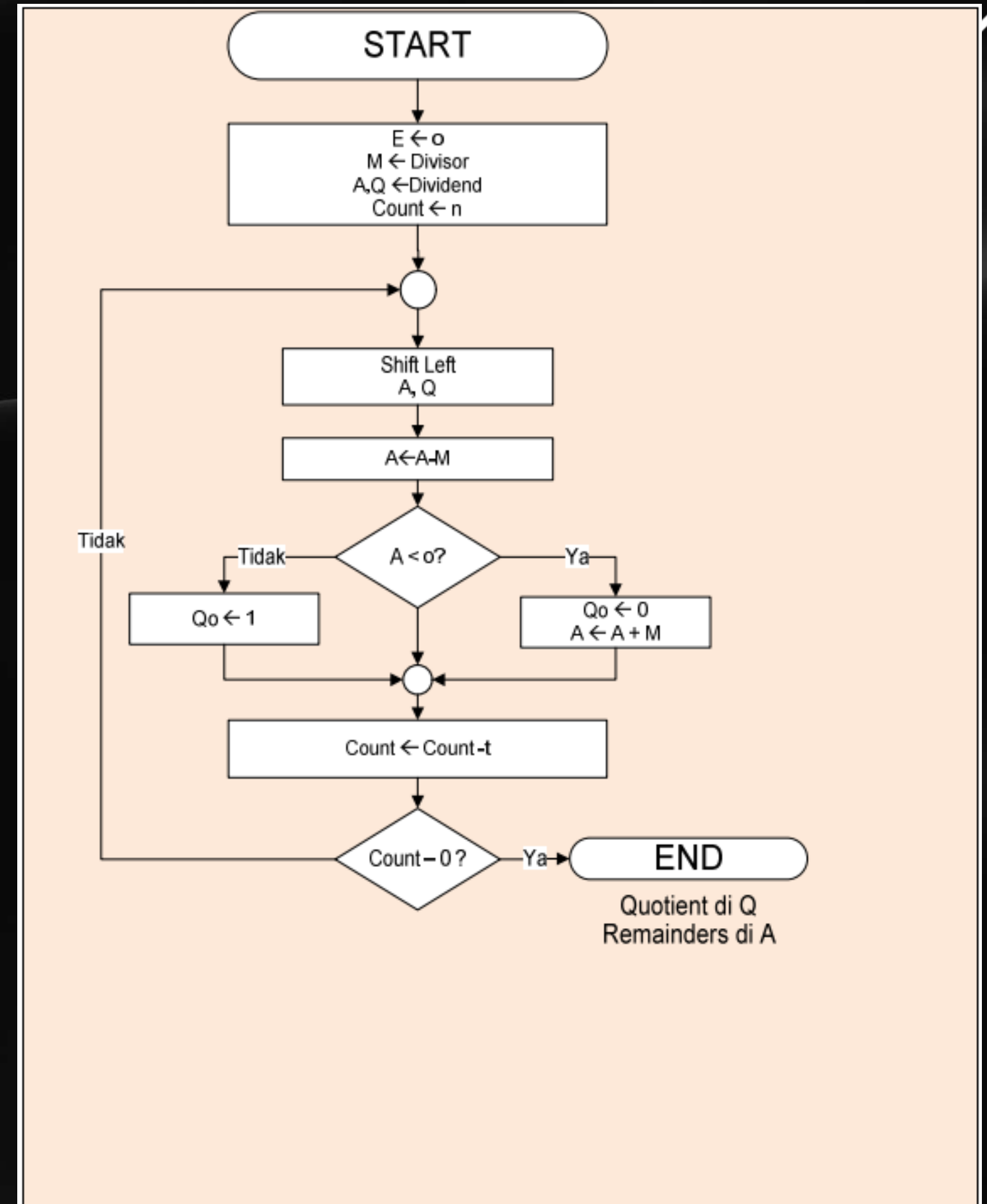
1. A dan Q1 diinisialisasi 0.
2. Control Logic memeriksa bit-bit multiplier satu-persatu beserta bit di kanannya.
3. Jika kedua bit sama (1-1 atau 0-0), maka seluruh bit di A, Q, dan Q1 digeser 1-bit ke kanan.
4. Jika kedua bit bernilai berbeda, ada 2 kondisi; multiplicand ditambahkan ke register A jika kedua bit bernilai (0-1); multiplicand dikurangkan ke register A jika kedua bit bernilai (1-0).
5. Proses tersebut dilakukan secara berulang untuk setiap bit multiplier.
6. Hasil perkalian akhir tersimpan di register A dan Q.

Secara singkat,
algoritmanya bisa
digambarkan sebagai
berikut :



PEMBAGIAN

Operasi pembagian memiliki dua operand yang diantaranya dividend dan divisor. Divident adalah bilangan yang dibagi, sedangkan divisor adalah bilangan pembagi. Flowchart berikut merupakan gambaran algoritma untuk melakukan operasi pembagian.



Kalkulasi biner dalam pembagian antara 147 dengan 11 dicontohkan seperti berikut :

147 = 10010011 → Dividend (A,Q)			
11 = 1011 → Divisor (M)			
-11 = 0101 (-M)			
E	A	Q	
0	1001	0011	Start
1	0010	0110	Geser kiri
	0101		A-M
1	0111		Hasil A-M
1	0111	0111	Set Q ₀
0	1110	1110	Geser kiri
	0101		A-M
1	0011		Hasil A-M
1	0011	1111	Set Q ₀
0	0111	1110	Geser kiri
	0101		A-M
0	1100		Hasil A-M
	1011		A+M
1	0111	1110	Hasil A+M
0	1111	1100	Geser kiri
	0101		A-M
1	0100		Hasil A-M
1	0100	<u>1101</u>	Set Q ₀

Masuk ke implementasi, berikut adalah contoh dari operasi perkalian antara 8 dan 4 menggunakan algoritma Booth :

8 : 1000 (multiplicand) → -8 : 1000			
4 : 0100 (multiplier) → Q			
A	Q	Q ₁	
0000	0100	<u>0</u>	Initial Value
0000	0010	<u>0</u>	Hasil geser kanan Siklus I
0000	0001	<u>0</u>	Hasil geser kanan Siklus II
1000	0001	<u>0</u>	Hasil pengurangan
<u>1100</u>	0000	<u>1</u>	Hasil geser kanan Siklus III
0100	0000	<u>1</u>	Hasil penjumlahan
<u>0010</u>	<u>0000</u>	<u>0</u>	Hasil geser kanan Siklus IV
Product perkalian antara 1000 dan 0100 adalah 0010 0000			

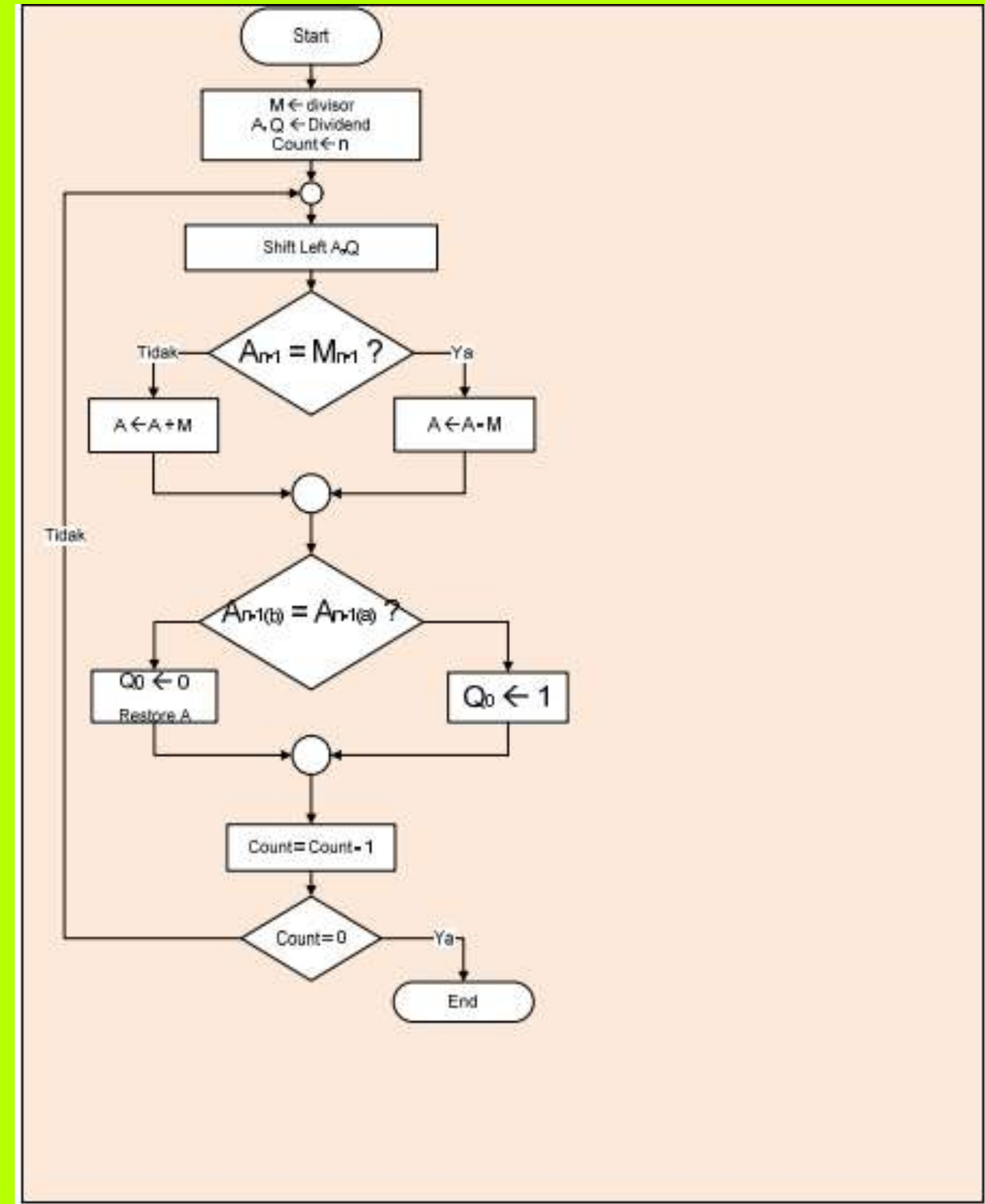
Masuk ke Coba perhatikan kembali contoh di atas, jika terjadi pengurangan oleh multiplicand terhadap register A, bit paling kiri hasil geser kanan 1-bit akan diisi dengan 1. Sebaliknya, jika tidak terjadi pengurangan maka bit paling kiri hasil geser kanan 1-bit akan diisi dengan biner 0. implementasi, berikut adalah contoh dari operasi perkalian antara 8 dan 4 menggunakan algoritma Booth :

PEMBAGIAN KOMPLEMEN-2

Langkah untuk melakukan pembagian dengan menggunakan metode komplement-2 adalah sebagai berikut :

1. Muatkan divisor ke M, dividend ke A dan Q. Divident diekspresikan sebagai komplement-2 2-nbit.
2. Geser A dan Q 1-bit ke kiri.
3. Bila M dan A memiliki tanda yang sama, lakukan $A \leftarrow A - M$; bila tandanya berbeda, $A \leftarrow A + M$
4. Operasi tersebut akan berhasil bila tanda A sesudah dan sebelum operasi sama
□ bila berhasil ($A \text{ dan } Q = 0$), set $Q_0 \leftarrow 1$ □ bila gagal ($A \text{ dan } Q \neq 0$), reset $Q_0 \leftarrow 0$ dan simpan A sebelumnya
5. Ulangi langkah 2 sampai 4 untuk setiap posisi bit di Q
6. Bila tanda divisor dan dividend sama maka quotient ada di Q, jika tidak quotient adalah komplement-2 dari Q.
7. Remainder ada di A.

Secara singkat,
algoritmanya bisa
digambarkan sebagai
berikut :



Contoh :

Pembagian antara (-7) dan 3 :			Pembagian antara 7 dan (-3) :		
A	Q	M = 0011	A	Q	M = 1101
0000	0111	Initial Value	0000	0111	Initial Value
0000	1110	Shift	0000	1110	Shift
1101		Subtract	1101		Add
0000	1110	Restore	0000	1110	Restore
0001	1100	Shift	0001	1100	Shift
1110		Subtract	1110		Add
0001	1100	Restore	0001	1100	Restore
0011	1000	Shift	0011	1000	Shift
0000		Subtract	0000		Add
0000	1001	Set $Q_0 = 1$	0000	1001	Set $Q_0 = 1$
0001	0010	Shift	0001	0010	Shift
1110		Subtract	1110		Add
0001	0010	Restore	0001	0010	Restore

Pembagian antara $(-7)/3$ dan $7/(-3)$ akan menghasilkan remainder yang berbeda. Hal ini disebabkan operasi pembagian didefinisikan sebagai berikut :

$$D = Q * V + R$$

Dengan

D = dividend

Q = quotient

V = divisor

R = remainder

THANK YOU

