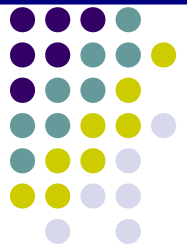
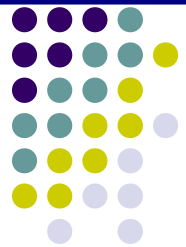


Arsitektur dan Organisasi Komputer

6-1

Aditya Wikan Mahastama, S.Kom



Week 9

Computer Arithmetic (1)
ALU dan Operasi Integer

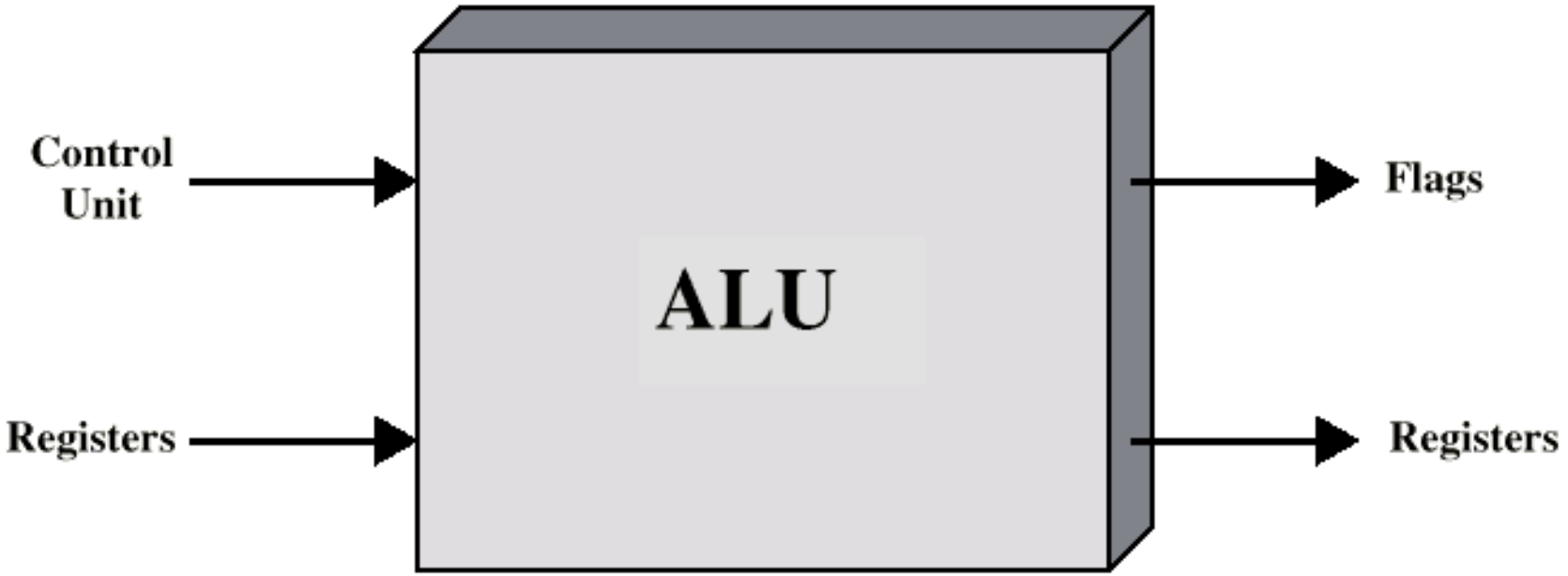
Arithmetic & Logic Unit

- Tugas ALU: Melakukan kalkulasi biner
- Semua komponen lain dalam komputer sebenarnya ada untuk melayani ALU
- ALU hanya bisa menangani bilangan bulat (integer)

However:

- ALU bisa juga menangani bilangan pecahan (real)
- Umumnya dalam bentuk FPU (Floating Point Unit) terpisah (maths co-processor)
- Co-processor dalam chip terpisah (486DX +)

Input dan Output ALU



Flags: menandai bagaimana sifat hasil kalkulasi.
Misal: hasil kalkulasi overflow, maka flag akan bernilai 1.
Nilai flag nantinya juga akan disimpan ke dalam register.

Representasi Integer

- Meskipun secara matematis, dalam sistem bilangan biner bisa digunakan tanda minus dan *radix point*, di dalam komputer hanya ada bilangan 0 & 1 untuk merepresentasikan semua angka
- Contoh bilangan biner matematis:

  -1101.0101 = -13.3125
  Bentuk seperti ini tidak membawa manfaat bagi komputer, malah menyulitkan

  Oleh karena itu tidak dipakai

Representasi Integer Positif

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

  00000000 = 0

  00000001 = 1

  00101001 = 41

  10000000 = 128

  11111111 = 255

  →tidak ada masalah!!

Representasi Integer Negatif (1)

- Mulai timbul masalah saat akan menyimpan bilangan negatif
- Komputer tidak mengenal tanda minus

Sign-Magnitude Representation

- Bit paling kiri menunjukkan magnitude integer (positif atau negatif) → sign bit
- +18 = **0**0010010

  - 18 = **1**0010010
- Kekurangan: ada 2 buah angka nol: nol positif (**0**0000000) dan negatif (**1**0000000)

Representasi Integer Negatif (2)

Two's Complement Representation

- Ini yang digunakan di komputer sekarang
- Satu bit paling kiri dijadikan bernilai negatif, kemudian dijumlahkan dengan bit sisanya. Nalar manusia paling gampang, gunakan Value Box:
- Misal 8-bit:

  -128 64 32 16 8 4 2 1

  Mengapa 1 bit paling kiri? Porsi sama besar:

  negatif = -128

  positif = (64 + 32 + 16 + 8 + 4 + 2 + 1) = 127

Aritmetik Integer - Negasi (1)

Two's Complement Operation

- Tentukan komplemen boolean untuk tiap bit, termasuk sign bitnya. Komplemen: 1 jadi 0, 0 jadi satu.
- Kemudian tambahkan 1 pada hasilnya
- Contoh: + 18 = 00010010

```
          11101101  (bitwise complement)
          +       1
          ----------
- 18 =    11101110
```

Aritmetik Integer - Negasi (1)

Two's Complement Operation

- Keuntungan: Hanya ada satu nol (coba negasikan 0).
- Terjadi ketimpangan representasi nilai negatif dan positif untuk jumlah bit tertentu

  misal untuk 8-bit, range bilangan bulat yang terwakili adalah:

  -128 … 127 (bukan 128 tapi 128 - 1)

  → Inilah yg terjadi pada komputer kita. Silakan cek di tipe data bahasa pemrograman apapun!

Konversi antara panjang bit yang berbeda

- Mungkin saja antar register tidak memiliki panjang bit yang sama, misall dari 8 ke 16
- Positive number pack with leading zeros

  +18 = 00010010

  +18 = 00000000 00010010
- Negative numbers pack with leading ones

  -18 = 10010010

  -18 = 11111111 10010010
- i.e. pack with MSB (sign bit)

Aritmetik Integer – Addition dan Substraction (1)

- Disini terlihat jelas keuntungan menggunakan sistem two's complement. Tidak perlu ada sirkuit pengurangan, hanya ada komplemen dan penjumlahan
- Perlakuan sama untuk penjumlahan maupun pengurangan: langsung DIJUMLAHKAN

  karena 7 – 2 sama dengan 7 + (-2)
- Panjang bit bilangan-bilangan yang dijumlahkan maupun hasilnya, harus sama

Aritmetik Integer – Addition dan Substraction (2)

Contoh untuk 4-bit:

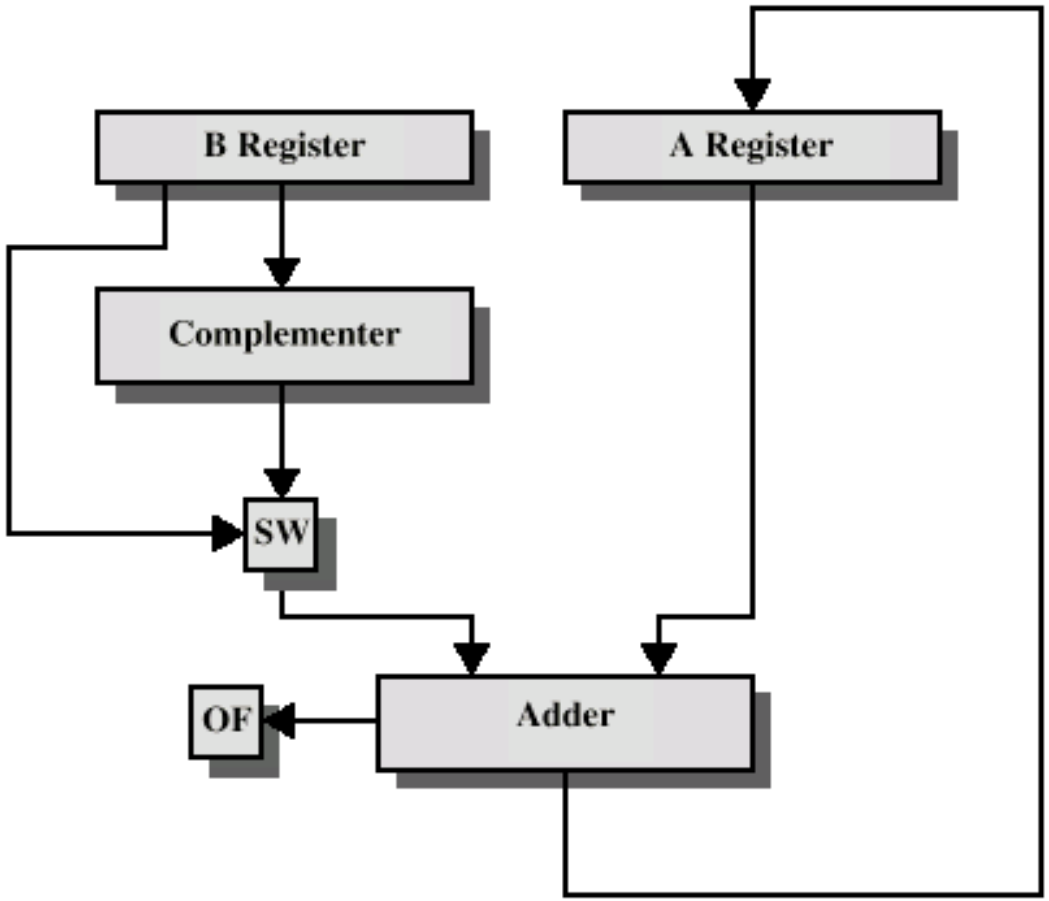
3 + 4	0011	= 3
	+0100	= 4
	0111	= 7
5 - 7	0101	= 5
	+1001	= -7
	1110	= -2

Latihan: -4 + 4, 4 – 1, 5 + 4, -7 – 6

Aritmetik Integer – Overflow

- Kadang penjumlahan tidak menghasilkan panjang digit yang sama, bisa lebih, kelebihan itu dihilangkan saja (dipotong).
- Overflow terjadi jika register ybs tidak mampu menampung bilangan yang dihasilkan. Misal 4-bit rangenya -8 … 7, tidak bisa menampung bilangan 11
- Overflow jika dan hanya jika: penjumlahan dilakukan terhadap dua bilangan bertanda sama, dan hasilnya bertanda berbeda

Addition dan Subtraction secara Hardware

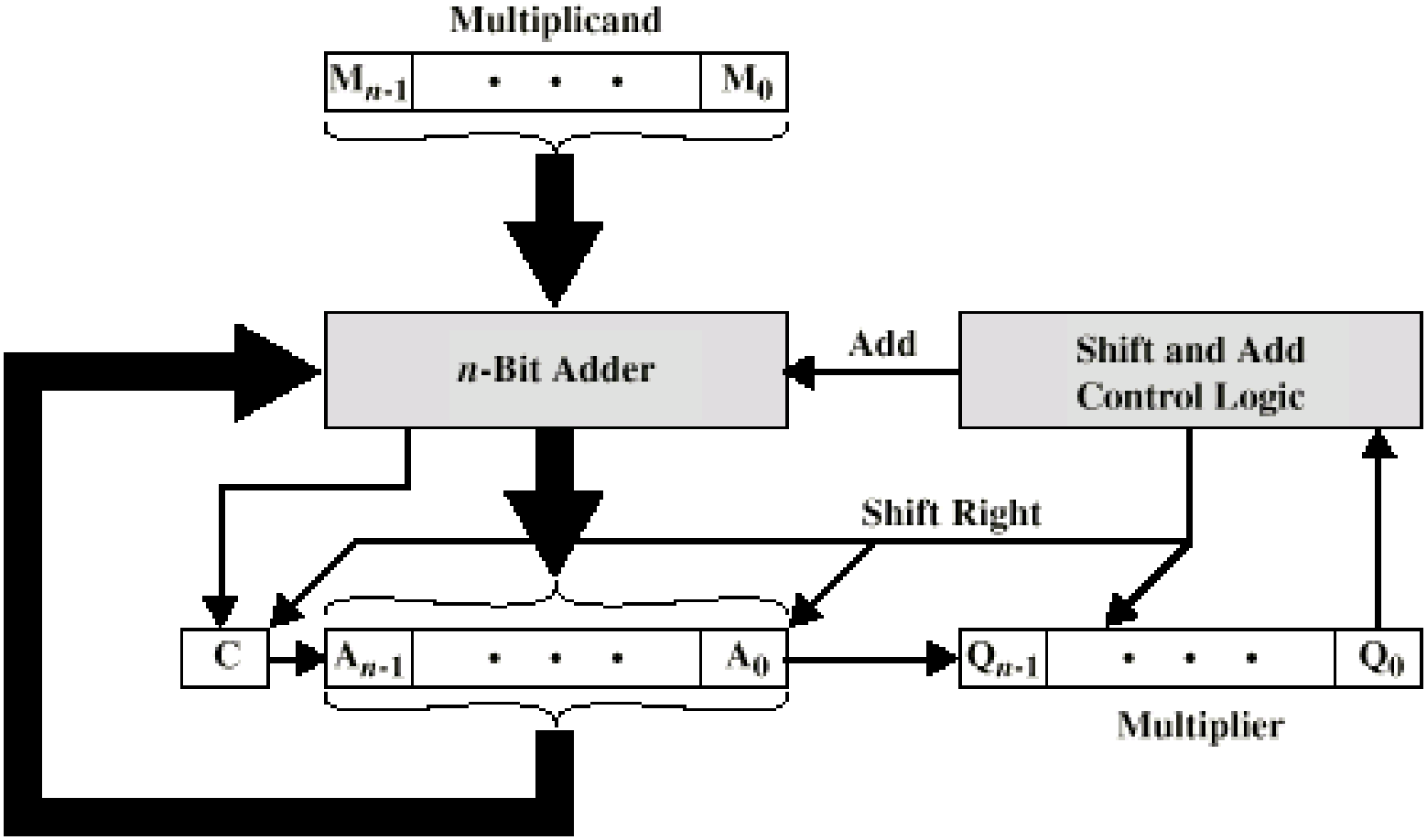


OF = overflow bit
SW = Switch (select addition or subtraction)

Aritmetik Integer – Multiplication (Perkalian)

- Untuk bilangan unsigned (positif semua), secara manusia perkalian dapat dilakukan secara manual (contoh di papan tulis)
- Tapi komputer tidak mempunyai tempat cukup untuk menyimpan *partial products* (hasil sementara) karena akan menghabiskan banyak tempat. Oleh karena itu digunakan metode geser (shift)

Aritmetik Integer – Multiplication
Unsigned Binary Multiplication (1)



(a) Block Diagram

Aritmetik Integer – Multiplication
Unsigned Binary Multiplication (2)

Aturan:

- Masukkan bilangan yang akan dikalikan ke M
- Masukkan bilangan pengali ke Q
- Awal: A dan C diset 0
- Jika kondisi terakhir Q_0 = 0, hanya lakukan shift kanan 1 bit sepanjang A dan Q, tapi
- Jika kondisi terakhir Q_0 = 1, tambahkan M ke A (jika tidak cukup, tampung sisa digitnya di C), kemudian lakukan shift kanan 1 bit sepanjang C, A dan Q

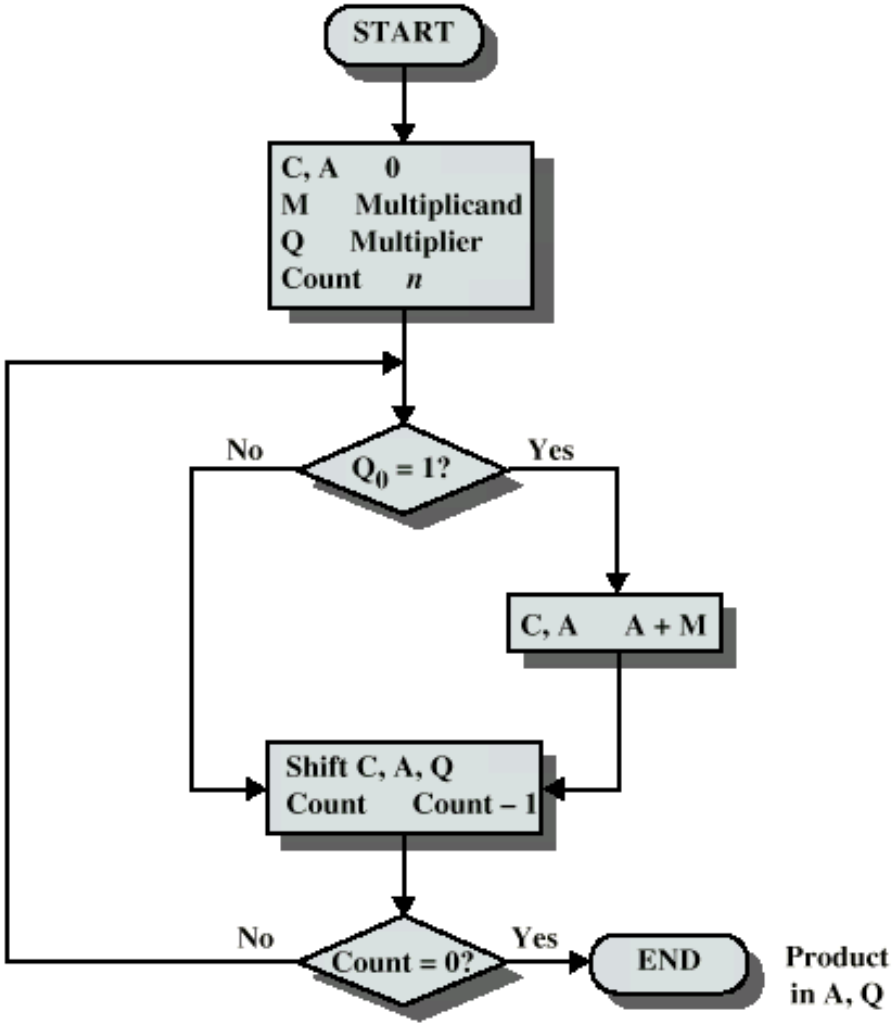
  Ulangi dua baris di atas sebanyak panjang bit Q
- Setelah loop terakhir, hasil akan tersedia di A dan Q

Aritmetik Integer – Multiplication
Unsigned Binary Multiplication (3)

Misal perkalian 1011 (11) x 1101 (13)

C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	

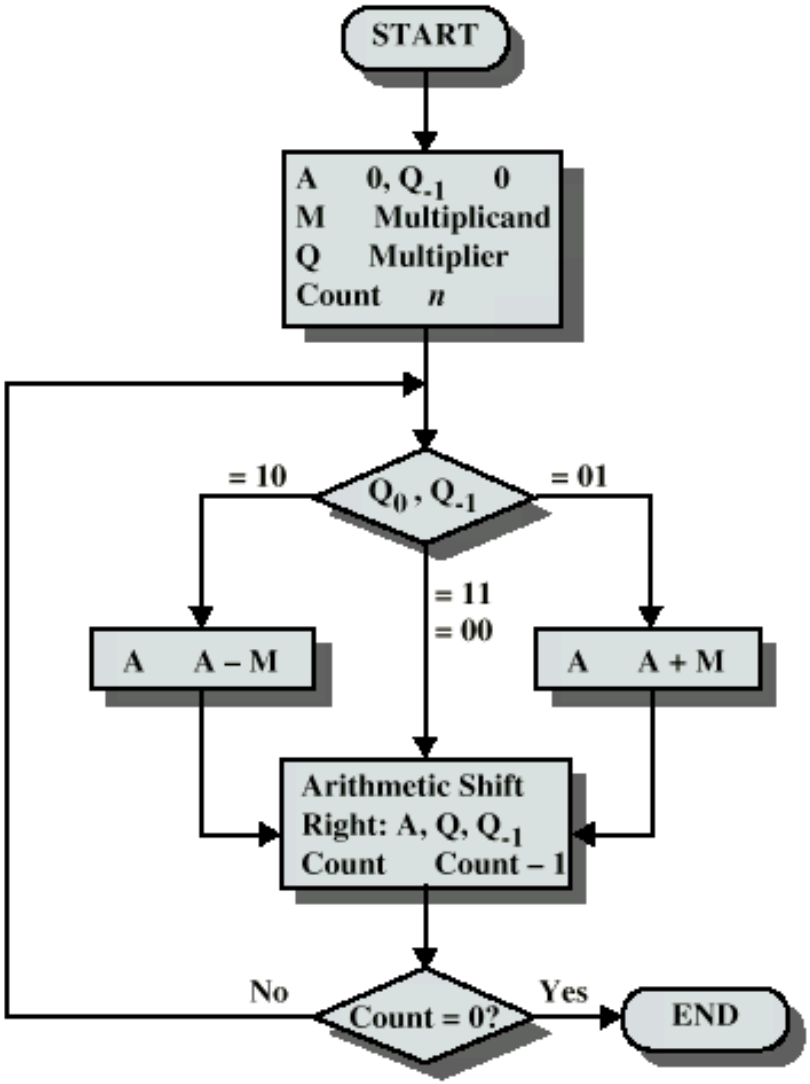
Aritmetik Integer – Multiplication
Unsigned Binary Multiplication (4)



Aritmetik Integer – Multiplication Perkalian Bilangan Negatif

- Silakan coba kalikan bilangan negatif dengan cara sebelumnya, pasti hasilnya aneh
- Solusi 1
  - Dasar: A X (-B) = - (A X B)
  - Positifkan bilangan yang negatif (dengan negasi terbalik)
  - Kalikan dengan cara sebelumnya
  - Kalau bilangan aslinya berbeda tanda (sign), negasikan hasilnya
- Solusi 2
  - Booth's algorithm

Aritmetik Integer – Multiplication Booth's Algorithm (1)



Aritmetik Integer – Division
Signed&Unsigned Binary Division (1)

Aturan:

- Masukkan pembagi ke register M
- Masukkan bilangan yang akan dibagi ke register A dan Q, dalam bentuk perpanjangan bit 2 kalinya.
- Shift kiri A dan Q sebanyak 1 bit.
- Jika M dan A bertanda sama, A ← A – M, kalau tidak A ← A + M
- Cek apakah tanda A berubah setelah operasi diatas
  Jika tanda sama atau hasil A=0, set Q0 ← 1
  Jika tanda beda dan hasil A≠0, set Q0 ← 0 dan pulihkan (restore) nilai A sebelumnya
- Ulangi ketiga baris di atas sebanyak panjang bit Q
- A akan berisi sisa, Q berisi hasil baginya

Aritmetik Integer – Division Signed&Unsigned Binary Division (2)