Langage machine

Jean-Luc Collinet
Loïc Lecharlier
José Vander Meulen

Année académique 2019-2020

Instructions NASM

MUL, SHL

MUL op

- MUL réalise une multiplication entière non signée de l'opérande op
- Le multiplicateur est l'op
- Le multiplicande est implicite selon la taille de l'op :
  - ✓ Cas où op est un byte : AL x op
  - ✓ Cas où op est un word : AX x op
  - ✓ Cas où op est un double word : EAX x op
- op peut être soit un registre soit un opérande en mémoire

MUL op

- Exemples avec des registres :
  - ✓ MUL BL
    - BL a une taille d'un byte
    - cela réalise AL x BL
  - ✓ MUL CX
    - CX a une taille d'un word
    - cela réalise AX x CX
  - ✓ MUL ESI
    - ESI a une taille d'un double word
    - cela réalise EAX x ESI

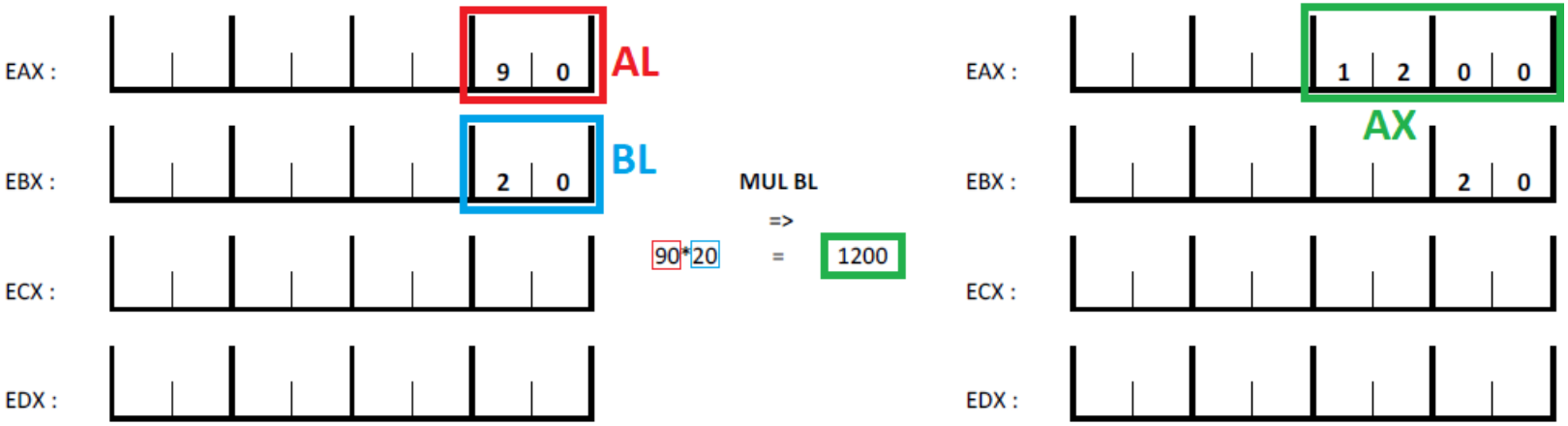
MUL op

- Là où va le résultat est implicite selon la taille de l'op :
  - ✓ Cas où op est un byte : AX
  - ✓ Cas où op est un word : DX_AX
  - ✓ Cas où op est un double word : EDX_EAX
- _ signifie "suivi de"

MUL op

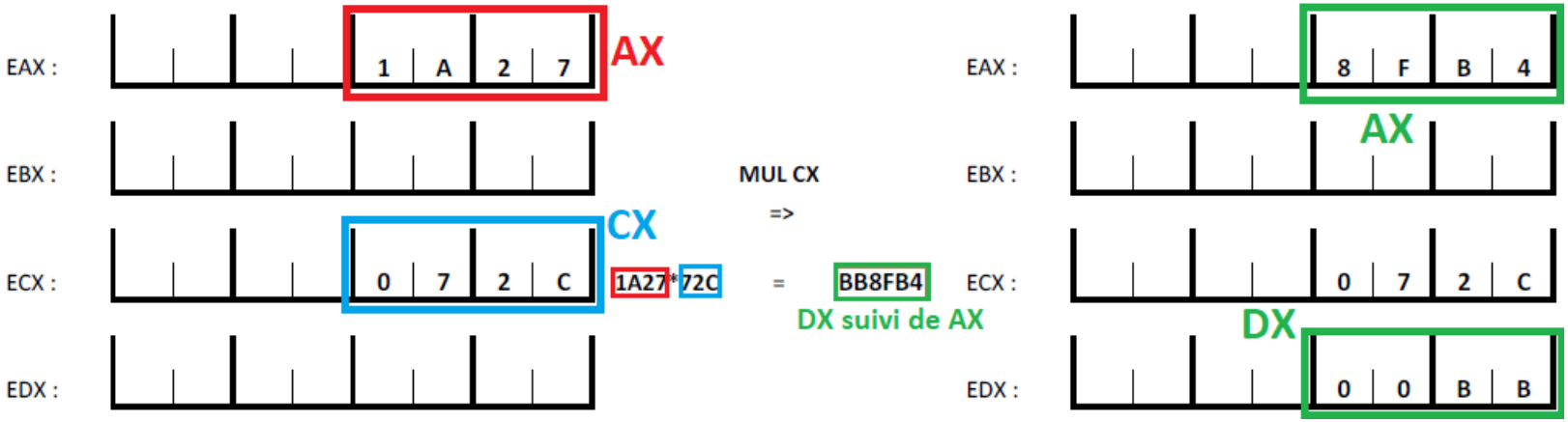
- Exemples avec des registres :
  - ✓ MUL BL
    - BL est un byte
    - AX reçoit le résultat de AL x BL
  - ✓ MUL CX
    - CX est un word
    - DX_AX reçoit le résultat de AX x CX
  - ✓ MUL ESI
    - ESI est un double word
    - EDX_EAX reçoit le résultat de EAX x ESI

MUL BL



AX = AL*BL

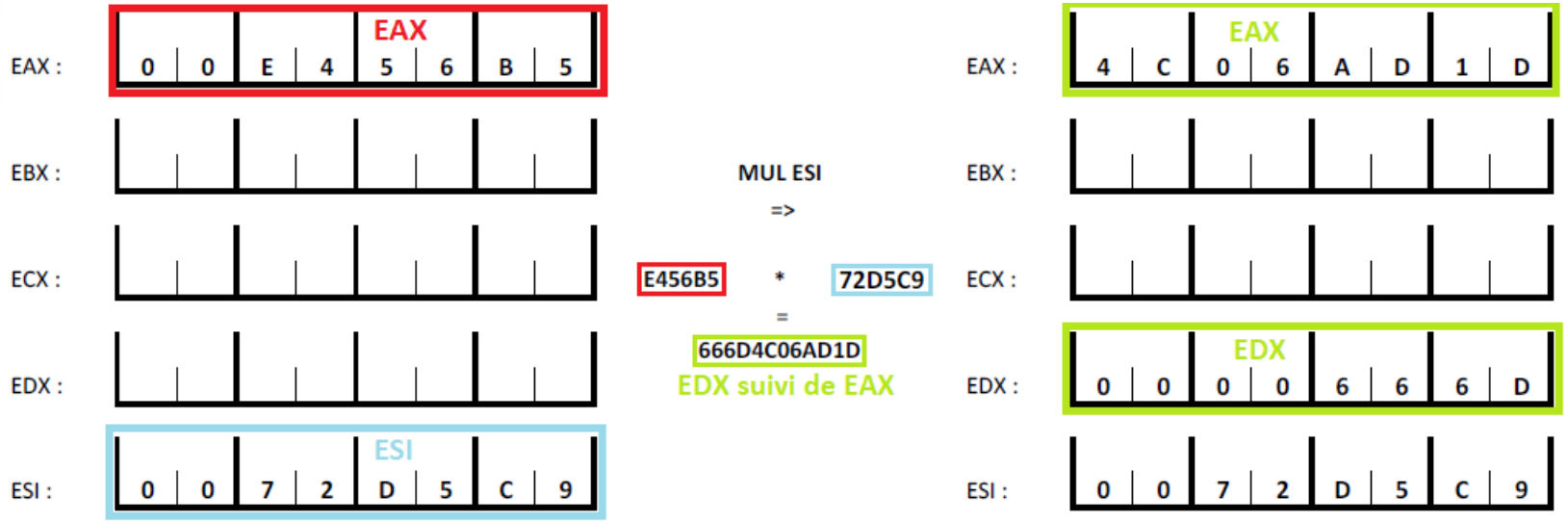
MUL CX



DX_AX = AX*CX

Concaténation : SHL EDX, 16 suivi de ADD EAX, EDX

MUL ESI



EDX_EAX = EAX * ESI

Attention ! : On est en 32 bits => concaténation impossible !

MUL : tableau synthèse

Implied operands:

Multiplicand	Multiplier	Product
AL	*r/m8*	AX
AX	*r/m16*	DX:AX
EAX	*r/m32*	EDX:EAX

SHL op,imm

- SHL réalise un décalage (*shift*) vers la gauche (*left*) des bits de l'op de imm position(s)
- Exemples :
  - ✓ SHL AL,1
    - Supposons que AL soit 01100110, AL devient après le *shift left* AL,1 : 11001100
  - ✓ SHL AL,3
    - Supposons que AL soit 01100110, AL devient après le *shift left* AL,3 : 00110000
  - ✓ On remplit à droite avec un ou des 0

SHL en pratique

- CL peut contenir la valeur de décalage
- SHL permet de multiplier par 2 en décalant d'une position vers la gauche

Instructions NASM

DIV, SHR

DIV op

- DIV réalise une division entière non signée par l'opérande op
- Le diviseur est l'op
- Le dividende est implicite selon la taille de l'op :
  - ✓ Cas où op est un byte : AX / op
  - ✓ Cas où op est un word : DX_AX / op
  - ✓ Cas où op est un dword : EDX_EAX / op
- op peut être soit un registre soit un opérande en mémoire

DIV op

- Exemples avec des registres :
  - ✓ DIV BL
    - BL a une taille d'un byte
    - cela réalise AX / BL
  - ✓ DIV CX
    - CX a une taille d'un word
    - cela réalise DX_AX / CX
  - ✓ DIV ESI
    - ESI a une taille d'un double word
    - cela réalise EDX_EAX / ESI

DIV op

- Là où vont le quotient et le reste est implicite selon la taille de l'op :
  - ✓ Cas où op est un byte :
    - Quotient : AL
    - Reste : AH
  - ✓ Cas où op est un word :
    - Quotient : AX
    - Reste : DX
  - ✓ Cas où op est un double word :
    - Quotient : EAX
    - Reste : EDX

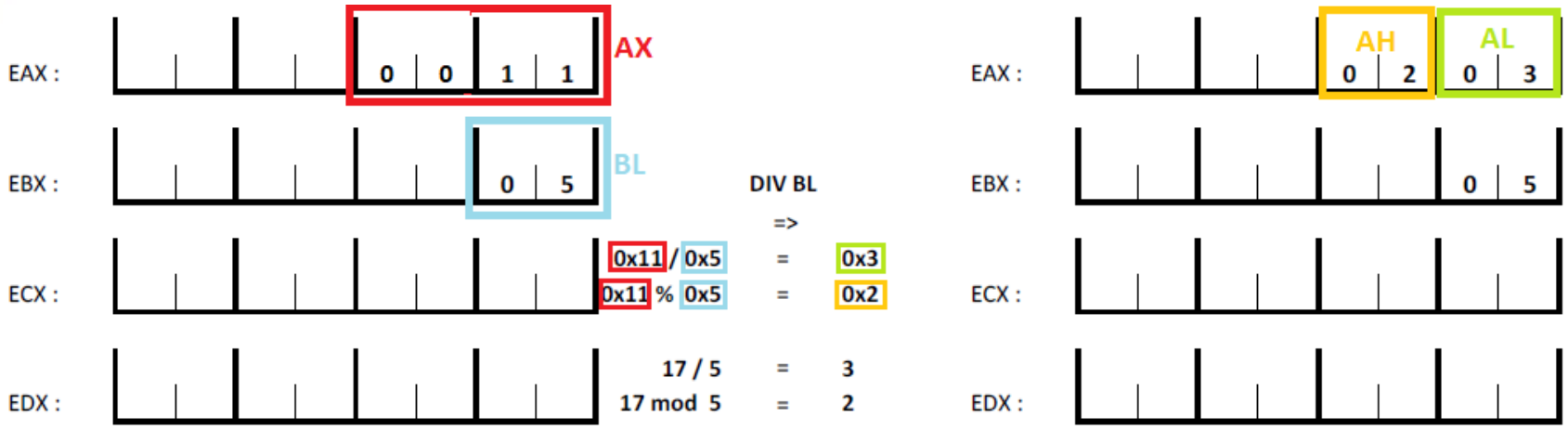
DIV op

- Exemples avec des registres :
  - ✓ DIV BL
    - BL est un byte
    - AL reçoit le quotient de AX / BL
    - AH reçoit le reste de AX / BL
  - ✓ DIV CX
    - CX est un word
    - AX reçoit le quotient de DX_AX / CX
    - DX reçoit le reste de DX_AX / CX

DIV op

- Exemples avec des registres :
  - ✓ DIV ESI
    - ESI est un double word
    - EAX reçoit le quotient de EDX_EAX / ESI
    - EDX reçoit le reste de EDX_EAX / ESI

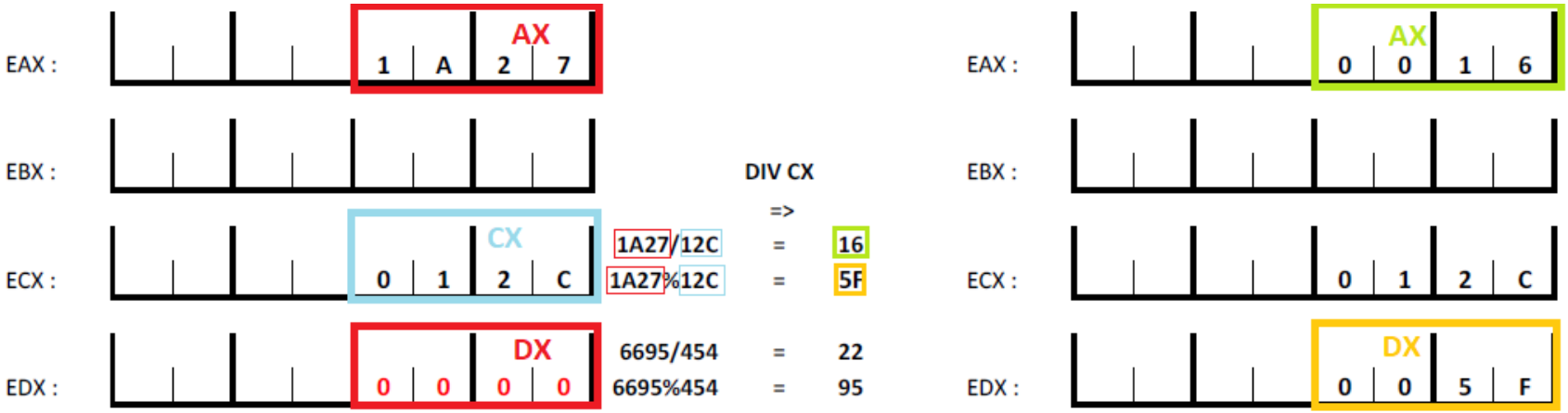
DIV BL



AL = AX / BL

AH = AX % BL

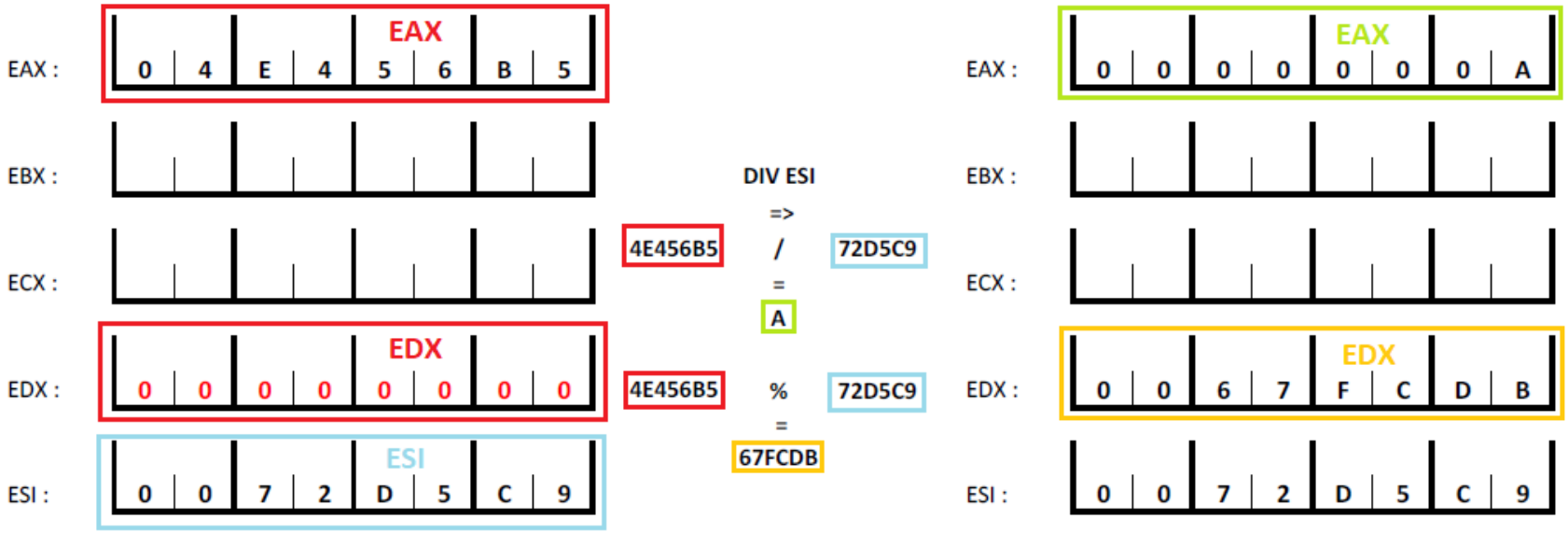
DIV CX



AX = DX_AX / CX

DX = DX_AX % CX

DIV ESI



EAX = EDX_EAX / ESI

EDX = EDX_EAX % ESI

DIV : tableau synthèse

Default Operands:

Dividend	Divisor	Quotient	Remainder
AX	*r/m8*	AL	AH
DX:AX	*r/m16*	AX	DX
EDX:EAX	*r/m32*	EAX	EDX

SHR op,imm

- SHR réalise un décalage (*shift*) vers la droite (right) des bits de l'op de imm position(s)
- Exemples :
  - ✓ SHR AL,1
    - Supposons que AL soit 01100110, AL devient après le *shift right* AL,1 : 00110011
  - ✓ SHR AL,3
    - Supposons que AL soit 01100110, AL devient après le *shift right* AL,3 : 00001100
  - ✓ On remplit à gauche avec un ou des 0

SHR en pratique

- CL peut contenir la valeur de décalage
- SHR permet de diviser par 2 en décalant d'une position vers la droite