Computer Organization and Assembly Language

Lab 9 notes

ADC Instruction

Example:

```
.data
op1 QWORD 0A2B2A40674981234h
op2 QWORD 08010870000234502h
sum DWORD 3 dup(?)
    ; = 0000000122C32B0674BB5736
.code
...
mov  esi,OFFSET op1 ; first operand
mov  edi,OFFSET op2 ; second operand
mov  ebx,OFFSET sum ; sum operand
mov  ecx,2          ; number of doublewords
call Extended_Add
...
```

Example

```
Extended_Add PROC
Pushad
clc

L1:
   mov eax,[esi] ; get the first integer
   adc eax,[edi] ; add the second integer
   pushfd        ; save the Carry flag
   mov [ebx],eax ; store partial sum
   add esi,4     ; advance all 3 pointers
   add edi,4
   add ebx,4
   popfd         ; restore the Carry flag
   loop L1       ; repeat the loop
   adc word ptr [ebx],0 ; add leftover carry

    popad
    ret
 Extended_Add ENDP
```

SBB Instruction

- The SBB (subtract with borrow) instruction subtracts source operand from a destination operand and then subtracts the carry flag from the destination.

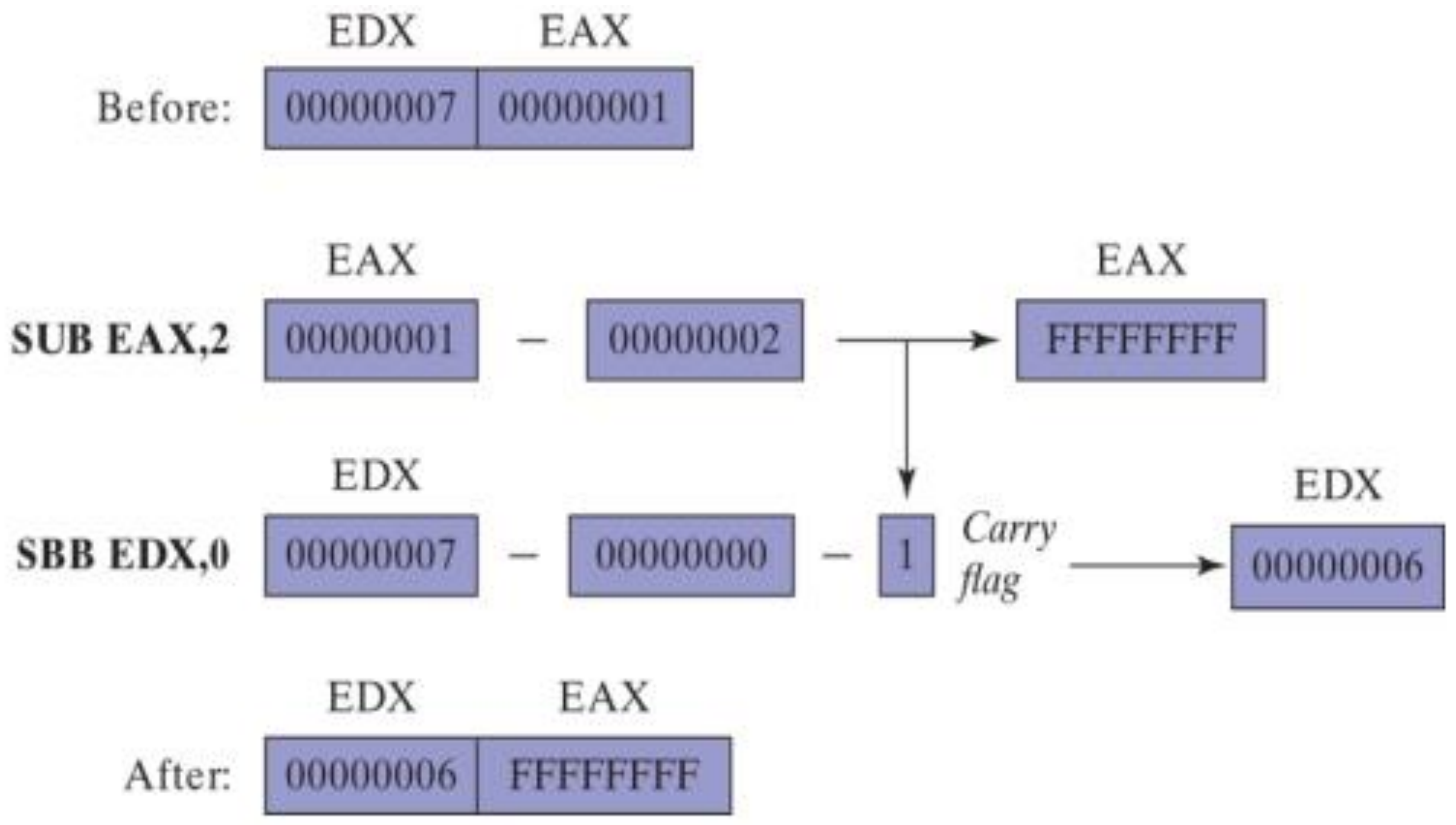
- The possible operands are same as for ADC instruction

Example:

- The following example code performs 64-bit subtraction. It sets EDX:EAX to 0000000700000001h and subtracts 1 from this value. The lower 32 bits are subtracted first, setting the Carry flag. Then the upper 32 bits are subtracted, including the Carry flag:

```
mov edx, 7        ; upper half
mov eax, 1        ; lower half
sub eax,  2       ; subtract 2
sbb edx, 0        ; subtract upper half
```

Steps:


EDX
EAX
Before:
00000007
00000001
EAX
SUB EAX,2
00000001
–
00000002
EAX
FFFFFFFF
EDX
SBB EDX,0
00000007
–
00000000
–
1
Carry flag
EDX
00000006
EDX
EAX
After:
00000006
FFFFFFFF

Implementing Arithmetic Expressions:

Exercise:

```
var4 = (var1 * -5) / (-var2 % var3);
```

```
mov   eax,var2        ; begin right side
neg   eax
cdq                   ; sign-extend dividend
idiv var3             ; EDX = remainder
mov   ebx,edx         ; EBX = right side
mov   eax,-5          ; begin left side
imul var1             ; EDX:EAX = left side
idiv ebx              ; final division
mov   var4,eax        ; quotient
```

Adapted from:

1. Assembly Language for x86 Processors by Kip R. Irvine (7th Edition)