W6-Arithmetic AL program

1.Create run.sh file
Terminal: nano run.sh

```
#!/bin/bash
nasm -f elf ./$1.asm
ld -m elf_i386 ./$1.o -o ./$1

./$1
```

2. Change Access permission for run.sh
Terminal: chmod 777 run.sh

3-1. Create file in Assembly Language code to run
Terminal : nano w6_1.asm
result = -var1 * 10
result = -5 * 10 = -50

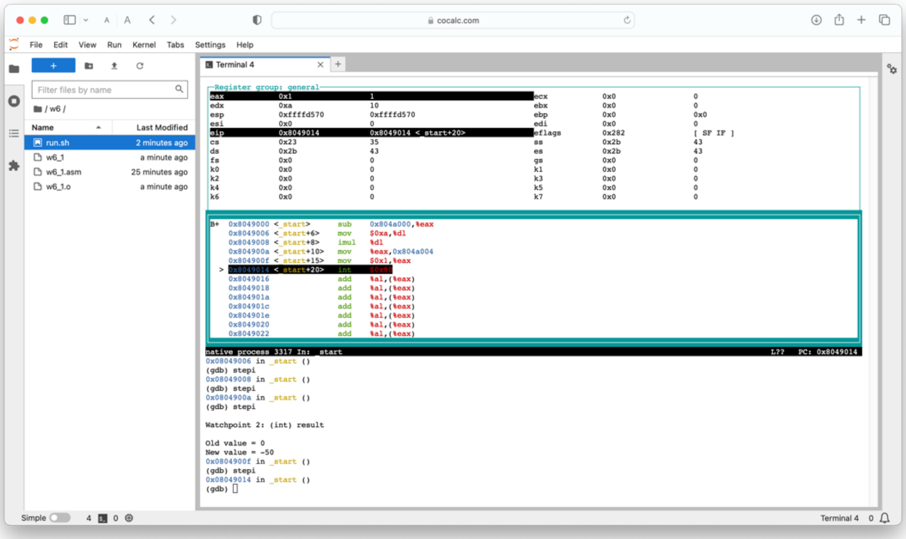
```
section .text
   global _start

_start:
   sub eax,[var1]        ;store -var1 into eax
   mov dl,10             ;store 10 into dl
   imul dl               ;multiply -var1 with 10
   mov [result],eax      ;store eax value into result variable

   mov eax,1
   int 0x80

section .data
   var1 DD 5            ;var1 is assigned 5

segment .bss

result resb 1           ;uninitialized variable
```

3-2. Run the result code with run.sh
Terminal: ./run.sh result

3-3. GDB debugging and checking register process

```
gdb result
layout asm
layout regs
watch (int) result
break _start
run
```

stepi <execute step by step.>



```
-Register group: general-
eax            0x1                 1                         ecx            0x0                 0
edx            0xa                 10                        ebx            0x0                 0
esp            0xffffd570          0xffffd570                ebp            0x0                 0x0
esi            0x0                 0                         edi            0x0                 0
eip            0x8049014           0x8049014 <_start+20>     eflags         0x282               [ SF IF ]
cs             0x23                35                        ss             0x2b                43
ds             0x2b                43                        es             0x2b                43
fs             0x0                 0                         gs             0x0                 0
k0             0x0                 0                         k1             0x0                 0
k2             0x0                 0                         k3             0x0                 0
k4             0x0                 0                         k5             0x0                 0
k6             0x0                 0                         k7             0x0                 0

B+  0x8049000 <_start>       sub    0x804a000,%eax
    0x8049006 <_start+6>     mov    $0xa,%dl
    0x8049008 <_start+8>     imul   %dl
    0x804900a <_start+10>    mov    %eax,0x804a004
    0x804900f <_start+15>    mov    $0x1,%eax
  > 0x8049014 <_start+20>    int    $0x80
    0x8049016                add    %al,(%eax)
    0x8049018                add    %al,(%eax)
    0x804901a                add    %al,(%eax)
    0x804901c                add    %al,(%eax)
    0x804901e                add    %al,(%eax)
    0x8049020                add    %al,(%eax)
    0x8049022                add    %al,(%eax)
native process 3317 In: _start                                                         L??   PC: 0x8049014
0x08049006 in _start ()
(gdb) stepi
0x08049008 in _start ()
(gdb) stepi
0x0804900a in _start ()
(gdb) stepi

Watchpoint 2: (int) result

Old value = 0
New value = -50
0x0804900f in _start ()
(gdb) stepi
0x08049014 in _start ()
(gdb) 
```

Watchpoint 2: (int) result

Old value = 0
New value =-50

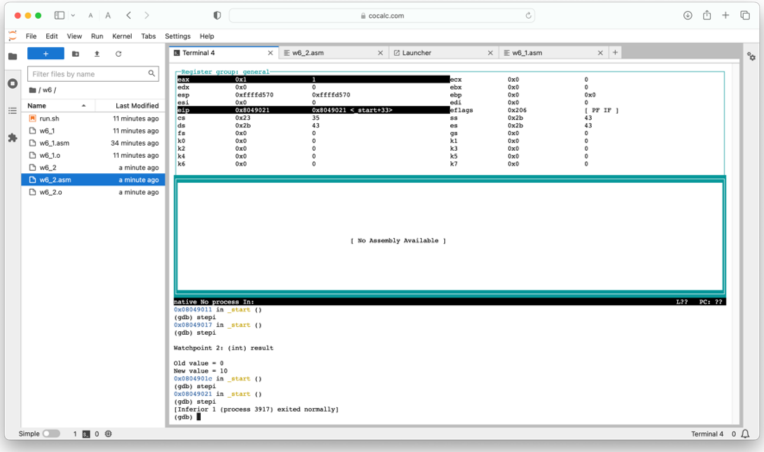
4-1. Create file in Assembly Language code to run
Terminal : nano w6_2.asm
result = var1 + var2 + var3 + var4
result = 1 + 2 + 3 + 4 = 10

```
section .text
    global _start

_start:
    mov eax,[var1]        ;store var1=1 into eax; eax is 1
    add eax,[var2]       ;add var2=2 to eax, eax is 3
    add eax,[var3]       ;add var3=3 to eax, eax is 6
    add eax,[var4]       ;add var4=4 to eax, eax is 10
    mov [result],eax     ;store eax=10 into result variable

    mov eax,1
    int 0x80

section .data
    var1 DD 1           ;var1 is assigned 1
    var2 DD 2           ;var1 is assigned 2
    var3 DD 3           ;var1 is assigned 3
    var4 DD 4           ;var1 is assigned 4

segment .bss
    result resb 1       ;uninitialized variable
```

4-2. Run the result code with run.sh
Terminal: ./run.sh result

4-3. GDB debugging and checking register process

gdb result
layout asm
layout regs
watch (int) result
break _start
run
stepi <execute step by step.>



Name	Last Modified
run.sh	11 minutes ago
w6_1	11 minutes ago
w6_1.asm	34 minutes ago
w6_1.o	11 minutes ago
w6_2	a minute ago
w6_2.asm	a minute ago
w6_2.o	a minute ago

```
-Register group: general-
eax            0x1                 1                          ecx            0x0                 0
edx            0x0                 0                          ebx            0x0                 0
esp            0xffffd570          0xffffd570                 ebp            0x0                 0x0
esi            0x0                 0                          edi            0x0                 0
eip            0x8049021           0x8049021 <_start+33>      eflags         0x206               [ PF IF ]
cs             0x23                35                         ss             0x2b                43
ds             0x2b                43                         es             0x2b                43
fs             0x0                 0                          gs             0x0                 0
k0             0x0                 0                          k1             0x0                 0
k2             0x0                 0                          k3             0x0                 0
k4             0x0                 0                          k5             0x0                 0
k6             0x0                 0                          k7             0x0                 0

                              [ No Assembly Available ]

native No process In:                                                                L??   PC: ??
0x08049011 in _start ()
(gdb) stepi
0x08049017 in _start ()
(gdb) stepi

Watchpoint 2: (int) result

Old value = 0
New value = 10
0x0804901c in _start ()
(gdb) stepi
0x08049021 in _start ()
(gdb) stepi
[Inferior 1 (process 3917) exited normally]
(gdb)
```

Watchpoint 2: (int) result

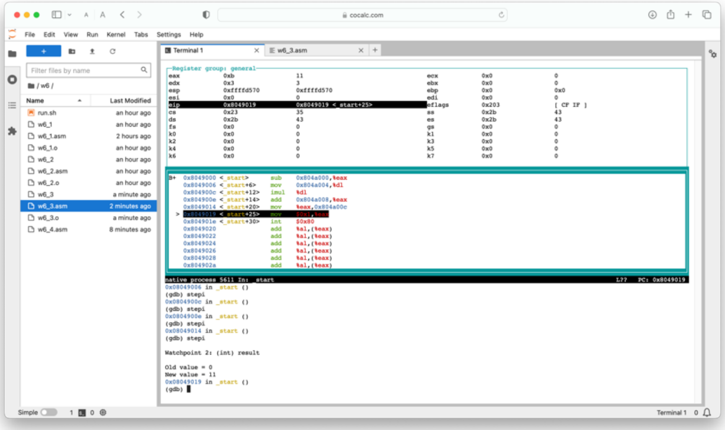
Old value = 0
New value =10

5-1. Create file in Assembly Language code to run
Terminal : nano w6_3.asm
result = (-var1 * var2) + var3
result = (-2 * 3) + 17 = 11

```
section .text
   global _start


_start:
   sub eax,[var1]        ;substitue eax=0 by var1=2 into eax; eax is -2
   mov dl, [var2]        ;store var2=3 into dl; dl is 3
   imul dl               ;multiply eax by dl=3, eax is -6
   add eax,[var3]        ;add var3=17 to eax, eax is 11
   mov [result],eax      ;store eax=11 into result variable

   mov eax,1
   int 0x80

section .data
   var1 DD 2             ;var1 is assigned 2
   var2 DD 3             ;var2 is assigned 3
   var3 DD 17            ;var3 is assigned 17

segment .bss
   result resb 1         ;uninitialized variable
```

5-2. Run the result code with run.sh
Terminal: ./run.sh result

5-3. GDB debugging and checking register process
gdb result
layout asm
layout regs
watch (int) result
break _start
run
stepi <execute step by step.>



```
-Register group: general-
eax            0xb                 11                          ecx            0x0                 0
edx            0x3                 3                           ebx            0x0                 0
esp            0xffffd570          0xffffd570                  ebp            0x0                 0x0
esi            0x0                 0                           edi            0x0                 0
eip            0x8049019           0x8049019 <_start+25>       eflags         0x203               [ CF IF ]
cs             0x23                35                          ss             0x2b                43
ds             0x2b                43                          es             0x2b                43
fs             0x0                 0                           gs             0x0                 0
k0             0x0                 0                           k1             0x0                 0
k2             0x0                 0                           k3             0x0                 0
k4             0x0                 0                           k5             0x0                 0
k6             0x0                 0                           k7             0x0                 0

B+ 0x8049000 <_start>       sub    0x804a000,%eax
   0x8049006 <_start+6>     mov    0x804a004,%dl
   0x804900c <_start+12>    imul   %dl
   0x804900e <_start+14>    add    0x804a008,%eax
   0x8049014 <_start+20>    mov    %eax,0x804a00c
 > 0x8049019 <_start+25>    mov    $0x1,%eax
   0x804901e <_start+30>    int    $0x80
   0x8049020                add    %al,(%eax)
   0x8049022                add    %al,(%eax)
   0x8049024                add    %al,(%eax)
   0x8049026                add    %al,(%eax)
   0x8049028                add    %al,(%eax)
   0x804902a                add    %al,(%eax)
native process 5611 In: _start                                              L??   PC: 0x8049019
0x08049006 in _start ()
(gdb) stepi
0x0804900c in _start ()
(gdb) stepi
0x0804900e in _start ()
(gdb) stepi
0x08049014 in _start ()
(gdb) stepi

Watchpoint 2: (int) result

Old value = 0
New value = 11
0x08049019 in _start ()
(gdb) 
```

Watchpoint 2: (int) result

Old value = 0
New value =11

6-1. Create file in Assembly Language code to run
Terminal : nano w6_4.asm
result = (var1 * 2)/(var2 - 3)
result = (10 * 2)/(8 - 3) = 4

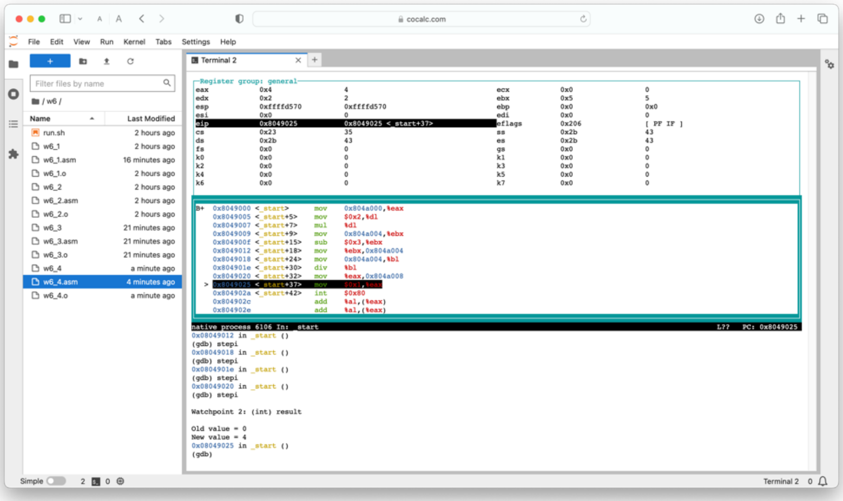
```
section .text
    global _start

_start:
    mov eax,[var1]        ;store var1=10 to eax; eax is 10
    mov dl, 2             ;store 2 into dl; dl is 2
    mul dl                ;multiply eax=10 by 2; eax is 20
    mov ebx,[var2]        ;store var2=8 to ebx; ebx is 8
    sub ebx,3             ;substitue ebx=8 by 3; ebx is 5
    mov [var2], ebx       ;return ebx=5 back to var2
    mov bl,[var2]        ;store ebx=5 as a divisor into bl
    div bl                ;divide eax=20 by bl=5; eax is 4
    mov [result],eax      ;store eax=4 into result variable

    mov eax,1
    int 0x80

section .data
```

```
    var1 DD 10          ;var1 is assigned 10
    var2 DD 8          ;var2 is assigned 8

segment .bss
    result resb 1        ;uninitialized variable
```

6-2. Run the result code with run.sh
Terminal: ./run.sh result

6-3. GDB debugging and checking register process

```
gdb result
layout asm
layout regs
watch (int) result
break _start
run
stepi <execute step by step.>
```



Name	Last Modified
run.sh	2 hours ago
w6_1	2 hours ago
w6_1.asm	16 minutes ago
w6_1.o	2 hours ago
w6_2	2 hours ago
w6_2.asm	2 hours ago
w6_2.o	2 hours ago
w6_3	21 minutes ago
w6_3.asm	21 minutes ago
w6_3.o	21 minutes ago
w6_4	a minute ago
w6_4.asm	4 minutes ago
w6_4.o	a minute ago

```
─Register group: general─
eax            0x4                 4                              ecx            0x0                 0
edx            0x2                 2                              ebx            0x5                 5
esp            0xffffd570          0xffffd570                     ebp            0x0                 0x0
esi            0x0                 0                              edi            0x0                 0
eip            0x8049025           0x8049025 <_start+37>          eflags         0x206               [ PF IF ]
cs             0x23                35                             ss             0x2b                43
ds             0x2b                43                             es             0x2b                43
fs             0x0                 0                              gs             0x0                 0
k0             0x0                 0                              k1             0x0                 0
k2             0x0                 0                              k3             0x0                 0
k4             0x0                 0                              k5             0x0                 0
k6             0x0                 0                              k7             0x0                 0

B+  0x8049000 <_start>       mov    0x804a000,%eax
    0x8049005 <_start+5>     mov    $0x2,%dl
    0x8049007 <_start+7>     mul    %dl
    0x8049009 <_start+9>     mov    0x804a004,%ebx
    0x804900f <_start+15>    sub    $0x3,%ebx
    0x8049012 <_start+18>    mov    %ebx,0x804a004
    0x8049018 <_start+24>    mov    0x804a004,%bl
    0x804901e <_start+30>    div    %bl
    0x8049020 <_start+32>    mov    %eax,0x804a008
  > 0x8049025 <_start+37>    mov    $0x1,%eax
    0x804902a <_start+42>    int    $0x80
    0x804902c                add    %al,(%eax)
    0x804902e                add    %al,(%eax)
native process 6106 In: _start                                                              L??   PC: 0x8049025
0x08049012 in _start ()
(gdb) stepi
0x08049018 in _start ()
(gdb) stepi
0x0804901e in _start ()
(gdb) stepi
0x08049020 in _start ()
(gdb) stepi

Watchpoint 2: (int) result

Old value = 0
New value = 4
0x08049025 in _start ()
(gdb)
```

Watchpoint 2: (int) result

Old value = 0
New value =4

***Challenge: For the arithmetic operation, I summarized following tips for this activity:

1. Use 'sub eax,[var1]' to get negative number of var1.
2. Operands for signed data (involving negative number) use imul/idiv instead of mul/div.
3. Operation on variables directly cause errors. It is necessary to put variable to register, do the substitution and then return the value back to the variable.
4. 'div ebx' get unexpected result. It is a better practice to introduce variable to dl or bl for multiplication and division.