# 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.>

A screenshot of a computer

Description automatically generated

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.>

A screenshot of a computer

Description automatically generated

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.>

A screenshot of a computer

Description automatically generated

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.>

A screenshot of a computer

Description automatically generated

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.