UNIX HW3 writeup

addsub1

```
mov eax, 0x9b92a550
add eax, 0x3b1ca0e0
sub eax, 0x30b83eb5
```

addsub2

```
mov eax, [0x600000]
add eax, [0x600004]
sub eax, [0x600008]
mov [0x60000c], eax
```

bubble

```
mov ecx, 9
outer:
push rcx
mov ecx, 9
mov esi, 0x600000

inner:
mov ebx, [esi]
mov edx, [esi+4]
cmp ebx, edx
jle pass
mov [esi], edx
mov [esi+4], ebx

pass:
add esi, 4
loop inner
pop rcx
loop outer
```

clear17

```
mov ebx, 0x20000
and ebx, eax
xor eax, ebx
```

dec2ascii

```
add al, 0x30
```

dispbin

```
mov ecx, 16

check:
mov bx, ax
and bx, 0x1
add bx, 0x30
mov BYTE PTR [0x600000 + ecx -1], bl
shr ax, 1
loop check
```

eval1

```
mov eax, [0x600000]
neg eax
add eax, [0x600004]
sub eax, [0x600008]
mov [0x60000c], eax
```

isolatebit

```
and ax, 0xfe0
shr ax, 5
mov BYTE PTR [0x600000], al
```

leax

```
mov esi, edi
imul esi, 0x2
mov eax, esi

xor esi, esi
mov esi, edi
imul esi, 0x3
mov ebx, esi

xor esi, esi
mov esi, edi
imul esi, 0x5
mov ecx, esi

xor esi, esi
mov esi, edi
imul esi, 0x9
mov edx, esi
```

loop15

```
xor ecx, ecx
.count:
cmp ecx, 0xf
jle .loop
jg  .exit

.loop:
mov al, BYTE PTR [0x600000 + ecx]
cmp al, 0
je .null
cmp al, 0x60
jle .upper
jg .lower

.inc:
inc ecx
jmp .count

.upper:
add al, 0x20
```

```
mov BYTE PTR [0x600010 + ecx], al
jmp .inc

.lower:
mov BYTE PTR [0x600010 + ecx], al
jmp .inc

.null:
mov BYTE PTR [0x600010 + ecx], al
jmp .inc

.exit:
```

math1

```
mov eax, [0x600000]
add eax, [0x600004]
imul eax, [0x600008]
mov [0x60000c], eax
```

math2

```
mov eax, [0x600000]
neg eax
imul eax, [0x600004]
add eax, [0x600008]
```

math3

```
mov eax, [0x600000]
mov ebx, [0x600004]
imul eax, 0x5
sub ebx, 0x3
idiv ebx
mov [0x600008], eax
```

math4

```
mov eax, [0x600000]; val1
imul eax, -5
mov ecx, [0x600004]
neg ecx
push rax
mov eax, ecx
xor rdx, rdx
cdq
idiv DWORD PTR [0x600008]
mov ecx, edx
pop rax
xor rdx, rdx
cdq
idiv ecx
mov [0x60000c], eax
```

math5

```
mov eax, [0x600000]       ;val1
mov esi, [0x600004]       ;val2
neg esi                   ;-val2
imul eax, esi             ; eax = val1 * -val2
mov esi, [0x600008]       ; val3
sub esi, ebx  ; esi = val3 - ebx
xor rdx, rdx
cdq
idiv esi
mov DWORD PTR [0x600008], eax
```

minicall

```
call   a
jmp    exit

a:
pop    rax
push   rax
ret

exit:
```

mulbyshift

```
mov rax, [0x600000]
shl rax, 5
sub rax, [0x600000]
sub rax, [0x600000]
sub rax, [0x600000]
sub rax, [0x600000]
sub rax, [0x600000]
sub rax, [0x600000]
mov [0x600004], rax
```

posneg

```
cmp eax, 0x0
jl .v1
xor esi, esi
inc esi
mov [0x600000], esi

.c2:
cmp ebx, 0x0
jl .v2
xor esi, esi
inc esi
mov [0x600004], esi

.c3:
cmp ecx, 0x0
jl .v3
xor esi, esi
inc esi
mov [0x600008], esi

.c4:
cmp edx, 0x0
jl .v4
xor esi, esi
inc esi
mov [0x600004], esi
jmp .end

.v1:
```

```
xor esi, esi
inc esi
neg esi
mov [0x600000], esi
jmp .c2

.v2:
xor esi, esi
inc esi
neg esi
mov [0x600004], esi
jmp .c3

.v3:
xor esi, esi
inc esi
neg esi
mov [0x600008], esi
jmp .c4

.v4:
xor esi, esi
inc esi
neg esi
mov [0x60000c], esi

.end:
```

recur

```
main:
mov edi, 0x13   ; depends on the question
call r
jmp .exit

r:
push    rbp
mov     rbp, rsp
push    rbx
sub     rsp, 24
mov     DWORD PTR [rbp-20], edi
cmp     DWORD PTR [rbp-20], 0
jg      .L2
mov     eax, 0
```

```
jmp     .L3

.L2:
cmp     DWORD PTR [rbp-20], 1
jne     .L4
mov     eax, 1
jmp     .L3

.L4:
mov     eax, DWORD PTR [rbp-20]
sub     eax, 1
mov     edi, eax
call    r
lea     rbx, [rax+rax]
mov     eax, DWORD PTR [rbp-20]
sub     eax, 2
mov     edi, eax
call    r
mov     rdx, rax
mov     rax, rdx
add     rax, rax
add     rax, rdx
add     rax, rbx

.L3:
add     rsp, 24
pop     rbx
pop     rbp
ret

.exit:
```

swapmem

```
mov rax, [0x600000]
mov rdx, [0x600008]
mov [0x600008], rax
mov [0x600000], rdx
```

swapreg

```
xchg rax, rbx
```

tolower

```
mov al, [0x600000]
sub al, 0x20
mov [0x600001], al
```

ul+lu

```
cmp ch, 0x60 ; ch>0x60 is upper
jg .upper
add ch, 0x20
jmp .end

.upper:
sub ch, 0x20

.end:
```