Introduction to Intel x86-64 Assembly, Architecture, Applications, & Alliteration

Xeno Kovah – 2014
xkovah at gmail
All materials is licensed under a Creative Commons “Share Alike” license.

- http://creativecommons.org/licenses/by-sa/3.0/

Attribution condition: You must indicate that derivative work
"Is derived from Xeno Kovah's 'Intro x86-64' class, available at http://OpenSecurityTraining.info/IntroX86-64-64.html"
Refresher - Boolean ("bitwise") logic

| AND | “&” | OR | “|” | XOR | “^” |
|-----|-----|----|----|-----|----|
| 0   | 0   | 0  | 0  | 0   | 0  |
| 0   | 1   | 0  | 1  | 1   | 1  |
| 1   | 0   | 0  | 1  | 1   | 1  |
| 1   | 1   | 1  | 1  | 1   | 0  |

Operands | Result
----------|--------

NOT “~”

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
AND - Logical AND

- Destination operand can be r/mX or register
- Source operand can be r/mX or register or immediate (No source *and* destination as r/mXs)

<table>
<thead>
<tr>
<th>and al, bl</th>
<th>and al, 0x42</th>
</tr>
</thead>
<tbody>
<tr>
<td>00110011b (al - 0x33)</td>
<td>00110011b (al - 0x33)</td>
</tr>
<tr>
<td>01010101b (bl - 0x55)</td>
<td>01000010b (imm - 0x42)</td>
</tr>
<tr>
<td>00010001b (al - 0x11)</td>
<td>00000010b (al - 0x02)</td>
</tr>
</tbody>
</table>

Book p. 231
OR - Logical Inclusive OR

- Destination operand can be r/mX or register
- Source operand can be r/mX or register or immediate (No source *and* destination as r/mXs)

<table>
<thead>
<tr>
<th>or al, bl</th>
<th>or al, 0x42</th>
</tr>
</thead>
<tbody>
<tr>
<td>00110011b (al - 0x33)</td>
<td>00110011b (al - 0x33)</td>
</tr>
<tr>
<td>OR 01010101b (bl - 0x55)</td>
<td>OR 01000010b (imm - 0x42)</td>
</tr>
<tr>
<td>result 01110111b (al - 0x77)</td>
<td>result 01110011b (al - 0x73)</td>
</tr>
</tbody>
</table>

Book p. 231
XOR - Logical Exclusive OR

- Destination operand can be r/mX or register
- Source operand can be r/mX or register or immediate (No source and destination as r/mXs)

xor al, al

<table>
<thead>
<tr>
<th>XOR</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>00110011b</td>
<td>00000000b</td>
</tr>
</tbody>
</table>

xor al, 0x42

<table>
<thead>
<tr>
<th>XOR</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>00110011b</td>
<td>01110001b</td>
</tr>
</tbody>
</table>

XOR is commonly used to zero a register, by XORing it with itself, because it's faster than a MOV

Book p. 231
NOT - One's Complement Negation

- Single source/destination operand can be r/mX

<table>
<thead>
<tr>
<th>NOT</th>
<th>result</th>
<th>NOT</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>00110011b (al - 0x33)</td>
<td>11001100b (al - 0xCC)</td>
<td>al</td>
<td>0x10000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bl</td>
<td>0x0001234</td>
</tr>
<tr>
<td></td>
<td></td>
<td>al+bl</td>
<td>0x10001234</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[al+bl]</td>
<td>0 (assumed memory at 0x10001234)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOT</td>
<td>00000000b</td>
</tr>
<tr>
<td></td>
<td></td>
<td>result</td>
<td>11111111b</td>
</tr>
</tbody>
</table>

Xeno trying to be clever on a boring example, and failing…

Book p. 231
#include <stdio.h>

int main()
{
    int i;
    for(i = 0; i < 10; i++)
    {
        printf("i = %d\n", i);
    }
}
INC/DEC - Increment / decrement

- Single source/destination operand can be r/mX
- Increase or decrease the value by 1
- When optimized, compilers will tend to favor not using inc/dec, as directed by the Intel optimization guide. So their presence may be indicative of hand-written, or un-optimized code.

Book p. 215-216
ForLoop.c - takeaways

- For loops will be some combination of a JCC (to determine if the exit condition is met yet), and an absolute JMP (to return from the end to the conditional checking code)
- In the absence of an explicit return value VS & GCC default to returning 0

```c
#include <stdio.h>

int main()
{
    int i;
    for(i = 0; i < 10; i++)
    {
        printf("i = %d\n", i);
    }
}
```

main:
```
000000140001010 sub    rsp,38h
000000140001014 mov     dword ptr [rsp+20h],0
00000014000101C jmp     0000000140001028
00000014000101E mov     eax,dword ptr [rsp+20h]
000000140001022 inc     eax
000000140001024 mov     dword ptr [rsp+20h],eax
000000140001028 cmp     dword ptr [rsp+20h],0Ah
00000014000102D jge     0000000140001042
00000014000102F mov     edx,dword ptr [rsp+20h]
000000140001033 lea     rcx,[40006000h]
00000014000103A call     qword ptr [40008368h]
000000140001040 jmp     000000014000101E
000000140001042 xor     eax,eax
000000140001044 add     rsp,38h
000000140001048 ret
```
Instructions we now know (22)

- NOP
- PUSH/POP
- CALL/RET
- MOV
- ADD/SUB
- IMUL
- MOVZX/MOVXSX
- LEA
- JMP/Jcc (family)
- CMP/TEST
- AND/OR/XOR/NOT
- INC/DEC