
318TH CYBERSP
SHADOW WARRIORS
90TH CYBERSPACE OPERATIONS SQ

Branching

Unconditional Jump

GAS/MASM/NASM

```
JMP LABEL     ; JMP doTheMath
```

```
top:
        mov al, 3
        add al, 5
        jmp bottom
middle:
        add al, 32
bottom:
        add al, 2
```

Conditional Jump

GAS

```
TESTS $L/M/%R, M/%R          # TESTB $32, %al
```

MASM

```
TEST M/R, L/M/R              ; TEST al, val
```

NASM

```
TEST SIZE [M]/R, L/[M]/R     ; TEST BYTE [val], ah
```

GAS

```
CMPS $L/M/%R, M/%R           # CMPB $32, %al
```

MASM

```
CMP M/R, L/M/R               ; CMP al, val
```

NASM

```
CMP SIZE [M]/R, L/[M]/R      ; CMP BYTE [val], ah
```

Conditional Jump

Table 5.7 Conditional jump instructions

Sign	Flag	Instruction	Description
Signed and unsigned	OF = 1	JO	Jump if overflow
	OF = 0	JNO	Jump if not overflow
	PF = 1	JP	Jump if parity
	PF = 0	JNP	Jump if not parity
	SF = 1	JS	Jump if sign
	SF = 0	JNS	Jump if not sign
	ZF = 1	JE	Jump if equal
		JZ	Jump if zero
	ZF = 0	JNE	Jump if not equal
		JNZ	Jump if not zero
	CX = 0	JCXZ	Jump if *CX* register is zero
	ECX = 0	JECXZ	Jump if *ECX* register is zero
	RCX = 0	JRCXZ	Jump if *RCX* register is zero

Conditional Jump continued

Signed	SF != OF	JL	Jump if less
		JNGE	Jump if not greater or equal
	SF = OF	JGE	Jump if greater or equal
		JNL	Jump if not less
	ZF = 1 or SF != OF	JLE	Jump if less or equal
		JNG	Jump if not greater
	ZF = 0 and SF = OF	JG	Jump if greater
		JNLE	Jump if not less or equal
Unsigned	CF = 1	JB	Jump if below
		JC	Jump if carry
		JNAE	Jump if not above or equal
	CF = 0	JAE	Jump if above or equal
		JNB	Jump if not below
		JNC	Jump if not carry
	CF = 1 or ZF = 1	JBE	Jump if below or equal
		JNA	Jump if not above
	CF = 0 and ZF = 0	JA	Jump if above
		JNBE	Jump if not below or equal

Compound Conditionals

```
if ( a > b && b > c)
      x = 1;
```

```
if ( a > b || b > c)
      x = 1;
```

Example 5.1 Compound conditionals (Intel syntax)

&& (Logical AND)	|| (Logical OR)

`cmp ax, bx  ; first expression`
`jbe next  ; quit if false`
`cmp bx, cx  ; second expression`
`jbe next  ; quit if false`
`mov x, 1  ; both are true`
`next:`
	`cmp ax, bx  ; first expression`
	`ja L1  ; if true skip to L1`
	`cmp bx, cx  ; second expression`
	`jbe next  ; false: skip to next`
	`L1: mov x, 1  ; true result`
	`next:`

Looping using CX/ECX/RCX

GAS/MASM/NASM

```
LOOP LABEL      ; LOOP sumLoop
```

Looping using CX/ECX/RCX

Example 5.2 Incorrect loop translation

C++	GAS	MASM/NASM

`int value = 0;`
`for (int x = 0; x < 2; x++)`
`  for (int y = 0; y < 3; y++)`
`  value++;`
	`movl $0, value`
	`movl $2, %ecx`
	`outer:`
	`  movl $3, %ecx`
	`  inner:`
	`  incl value`
	`  loop inner`
	`  loop outer`
		`mov value, 0`
		`mov ecx, 2`
		`outer:`
		`  mov ecx, 3`
		`  inner:`
		`  inc value`
		`  loop inner`
		`  loop outer`

Example 5.3 Correct loop translation

C++	GAS	MASM/NASM

`for (int x = 0; x < 2; x++)`
`  for (int y = 0; y < 3; y++)`
`  value++;`
	`movl $2, %ecx`
	`outer:`
	`  movl %ecx, counter`
	`  movl $3, %ecx`
	`  inner:`
	`  incl value`
	`  loop inner`
	`  movl counter, %ecx`
	`  loop outer`
		`mov ecx, 2`
		`outer:`
		`  mov counter, ecx`
		`  mov ecx, 3`
		`  inner:`
		`  inc value`
		`  loop inner`
		`  mov ecx, counter`
		`  loop outer`

Programmer-defined Counters

Example 5.4 Loop translation using CMP

C++	GAS	MASM/NASM

for (int x = 0; x < 2; x++)
  for (int y = 0; y < 3; y++)
  value++;
	movl $2, x
	outer:
	  movl $3, y
	  inner:
	  incl value
	  decl y
	  cmpl $0, y
	  jne inner
	  decl x
	  cmpl $0, x
	  jne outer
		mov x, 2
		outer:
		  mov y, 3
		  inner:
		  inc value
		  dec y
		  cmp y, 0
		  jne inner
		  dec x
		  cmp x, 0
		  jne outer

Using Registers

Example 5.5 Optimized loop translation

GAS	MASM/NASM

`xorl %eax, %eax`
`movl $2, %ebx`
`outer:`
`  movl $3, %ecx`
`  inner:`
`  incl %eax`
`  decl %ecx`
`  cmpl $0, %ecx`
`  jne inner`
`  decl %ebx`
`  cmpl $0, %ebx`
`  jne outer`
	`xor eax, eax`
	`mov ebx, 2`
	`outer:`
	`  mov ecx, 3`
	`  inner:`
	`  inc eax`
	`  dec ecx`
	`  cmp ecx, 0`
	`  jne inner`
	`  dec ebx`
	`  cmp ebx, 0`
	`  jne outer`

Translating While Loop

Example 5.6 `while` loop translations

C++	GAS v1	NASM/MASM v1	GAS v2	NASM/MASM v2

`x = 30;`
`while (x < 50)`
`  x++;`
	`movl $30, %eax`
	`while_loop:`
	`  cmpl $50, %eax`
	`  jae done`
	`  incl %eax`
	`  jmp while_loop`
	`done:`
		`mov eax, 30`
		`while_loop:`
		`  cmp eax, 50`
		`  jae done`
		`  inc eax`
		`  jmp while_loop`
		`done:`
			`movl $30, %eax`
			`while_loop:`
			`  cmpl $50, %eax`
			`  jb addone`
			`  jae done`
			`addone:`
			`  incl %eax`
			`  jmp while_loop`
			`done:`
				`mov eax, 30`
				`while_loop:`
				`  cmp eax, 50`
				`  jb addone`
				`  jae done`
				`addone:`
				`  inc eax`
				`  jmp while_loop`
				`done:`

Nested For Loop

Program 5.3 – *Nested* `for` *loop (64-bit)*

GAS	MASM	NASM

`.text`
`.globl _main`
`_main:`
 	
`xorq %rax, %rax`
`movq $2, %rbx`
`outer:`
`  movq $3, %rcx`
`  inner:`
`  incq %rax`
`  decq %rcx`
`  cmpq $0, %rcx`
`  jne inner`
`  decq %rbx`
`  cmpq $0, %rbx`
`  jne outer`
 	
`movq $0x2000001, %rax`
`movq $0, %rdi`
`syscall`
`.end`
	`extrn ExitProcess : proc`
	
	`.code`
	`_main PROC`
	
	`xor rax, rax`
	`mov rbx, 2`
	`outer:`
	`  mov rcx, 3`
	`  inner:`
	`  inc rax`
	`  dec rcx`
	`  cmp rcx, 0`
	`  jne inner`
	`  dec rbx`
	`  cmp rbx, 0`
	`  jne outer`
	
	`mov rcx, 0`
	`call ExitProcess`
	`_main ENDP`
	`END`
		`SECTION .text`
		`global _main`
		`_main:`
		
		`xor rax, rax`
		`mov rbx, 2`
		`outer:`
		`  mov rcx, 3`
		`  inner:`
		`  inc rax`
		`  dec rcx`
		`  cmp rcx, 0`
		`  jne inner`
		`  dec rbx`
		`  cmp rbx, 0`
		`  jne outer`
		
		`mov rax, 60`
		`xor rdi, rdi`
		`syscall`