Assembly Syntax Translation

Brian R. Hall
Department of Software Technology
Champlain College
Burlington, VT, USA
hall@champlain.edu

Refer to this document at your own risk. May contain errors. Scream if you wish, but no one will hear you.

- GAS prefixes registers with %
- GAS prefixes immediate values with $
- GAS also uses the $ prefix to indicate an address of a variable
- NASM and MASM use $ as the *current location counter*, while GAS uses the dot (.)
- GAS is source first, destination second
- NASM and MASM are destination first, source second
- GAS denotes operand sizes with *b*, *w*, *l*, and *q* suffixes on the instruction
- GAS and NASM labels are case-sensitive
- MASM labels are not case-sensitive
- GAS and NASM write FPU registers as ST0, ST1, etc.
- MASM writes FPU registers as ST(0), ST(1), etc.
- MASM relies more on assumptions (e.g., types), so sometimes it can be hard to tell what an instruction does
- GAS uses .equ to set a symbol to an expression, NASM uses the EQU directive, and MASM uses = or EQU
- All assemblers can use single or double quotes for strings.

Operation	**GAS**	**NASM**	**MASM**
Clear eax	xorl %eax, %eax	xor eax, eax	
Move contents of eax to esi	movl %eax, %esi	mov esi, eax	
Move contents of ax to si	movw %ax, %si	mov si, ax	
Move immediate byte value 4 to al	movb $4, %al	mov al, 4	
Move contents of address 0xf into eax	movl 0x0f, %eax	mov eax, [0x0f]	mov eax, ds:[0fh]
Move contents of variable temp into eax	movl temp, %eax	mov eax, DWORD [temp]	mov eax, temp
Move address of variable temp into eax	movl $temp, %eax	mov eax, temp	mov eax, OFFSET temp
Move immediate byte value 2 into temp	movl $2, temp	mov BYTE [temp], 2	mov [temp], 2
Move immediate byte value 2 into memory pointed to by eax	movb $2, (%eax)	mov BYTE [eax], 2	mov BYTE PTR [eax], 2
Move immediate word value 4 into memory pointed to by eax	movw $4, (%eax)	mov WORD [eax], 4	mov WORD PTR [eax], 4
Move immediate doubleword value 6 into memory pointed to by eax	movl $6, (%eax)	mov DWORD [eax], 6	mov DWORD PTR [eax], 6
Include syntax	.include “file.ext”	%include “file.ext”	INCLUDE file.ext
Label[1] syntax	label: type value		label type value
Current location counter	aSize: .long (. - array)[2]	aSize: EQU ($ - array)	aSize = ($ - array)

Operation	**GAS**	**NASM**	**MASM**
Reserve 64 bytes of memory	.space 64	resb 64	db 64 DUP (?)
Create uninitialized 32-bit variable temp	.lcomm temp, 4	temp: resd 1	temp DWORD ?
Create initialized 32-bit variable temp with value 5	temp: .long 5	temp: dd 5	temp DWORD 5
Create array w/ 32-bit values	temp: .long 5, 10, 15	temp: dd 5, 10, 15	temp DWORD 5, 10, 15
Create Hello World string	label: .ascii “Hello, World”	label: db ‘Hello, World’	label BYTE “Hello, World”
Create Hello World w/ newline and null terminated string	label: .asciz “Hello, World\n”	label: db ‘Hello, World’, 10, 0	label BYTE “Hello, World”, 10, 0
Procedure structure	label: ... ret		label PROC ... ret label ENDP
Program segments (sections)	.data .bss .text	SECTION .data SECTION .bss SECTION .text	.data .code
Types	.byte .word .long .quad	db dw dd dq	BYTE WORD DWORD QWORD
Repetition	label: .fill count, size, value	label: TIMES count type value	label type count DUP (value)
Macros	.macro label arg1, arg2 ... .endm	%macro label argcount ...args referenced as %1, %2 %endmacro	label MACRO arg1, arg2 ... ENDM

[1] Variable/identifier, not to be confused with MASM's LABEL directive
[2] aSize: .long (. - array) ; returns length in bytes
aSize: .long = (. - array) ; returns number of elements