Programmer's Reference to HLA Assembly Language

Typical Program Structure

```
    program progID;
    #include( "stdlib.hhf" );
    static
       variable declarations
    begin progID;
       statements
    end progID;
```

Assembly Language Instructions

Instruction	*Syntax*	*Description*
MOV	mov(source, dest);	dest = source;
ADD	add(source, dest);	dest += source;
SUB	sub(source, dest);	dest -= source;
SHL	shl(count, dest);	shuffles left a total of count bits in dest operand; sets carry when count=1
SHR	shr(count, dest);	shuffles right a total of count bits in dest operand; sets carry when count=1
SAR	sar(count, dest);	shuffles right a total of count bits in dest operand; sets carry when count=1; leaves H.O. bit unchanged
ROL	rol(count, dest);	rotates left a total of count bits in

Available Datatypes

int8
int16
int32

uns8
uns16
uns32

boolean

Available I/O Routines

stdout.put
stdout.puti8
stdout.puti16
stdout.puti32
stdout.putb
stdout.putw
stdout.putd
stdout.putu8
stdout.putu16
stdout.putu32

stdout.newln

stdin.get
stdin.geti8
stdin.geti16
stdin.geti32
stdin.getu8
stdin.getu16
stdin.getu32
stdin.getb
stdi n.getw
stdin.getd

		dest operand; sets carry when count=1
ROR	ror(count, dest);	rotates right a total of count bits in dest operand; sets carry when count=1
NOT	not(dest);	inverts the bits of the dest operand
AND	and(source, dest);	bitwise logical AND; result placed in dest operand
OR	or(source, dest);	bitwise inclusive OR; result placed in dest operand
XOR	xor(source, dest);	bitwise exclusive OR; result placed in dest operand
LAHF	lahf( );	pushes the lower 8 bits of EFLAGS register into AH
INC	inc(operand);	operand = operand + 1;
DEC	dec(operand);	operand = operand - 1;
CMP	cmp(lhs, rhs);	sets EFLAGS as if lhs-rhs was performed; does not change the value of

		either operand	
TEST	test(operand1, operand2);	sets EFLAGS as if AND(operand1, operand2) was performed; does not change the value of either operand	
NEG		neg(dest);	dest = - dest;

JMP	jmp label;
	jmp(32bit_register);
	jmp(dword);
		unconditional transfer of control. Note the inconsistent use of parentheses.	

SETcc	setcc(8bit_operand);	reads an EFLAG bit into a byte operand. Mnemonics listed below.	
Jcc	jcc label;	transfers control to label when condition is met. Mnemonics listed below.	

Mnemonics For SETcc and Jcc Instructions

Abbreviation	*Meaning*	*Example*
C	Set if Carry = 1	SETC
NC	Set if Carry = 0	SETNC
Z	Set if Zero = 1	SETZ
NZ	Set if Zero = 0	SETNZ

S	Set if Sign = 1	SETS
NS	Set if Sign = 0	SETNS
O	Set if Overflow = 1	SETO
NO	Set if Overflow = 0	SETNO
E	Set if Equal	SETE
NE	Set if Not Equal	SETNE
NA	Set if not >	SETNA
BE	Set if <=	SETBE
NAE	Set if not >=	SETNAE
B	Set if <	SETB
NB	Set if not <	SETNB
NBE	Set if not <=	SETNBE
A	Set if >	SETA
AE	Set if >=	SETAE
G	Set if greater than	SETG
NLE	Set if not less than or equal	SETNLE
GE	Set if greater than or equal	SETGE
NL	Set if not less than	SETNE
L	Set if less than	SETL
NGE	Set if not greater than or equal	SETNGE
LE	Set if less than or equal	SETLE
NG	Set if not greater than	SETNG