Intel Instruction Set (gas)

- These slides provide the gas format for a subset of the Intel processor instruction set, including:
  - Operation Mnemonic
  - Name of Operation
  - Syntax
  - Operation
  - Legal Operands
  - Examples
  - Description
  - Effect on Flag Bits

Gas Addressing Mode Syntax

- In this document, we use the gas syntax for each of these addressing modes:

– Register	reg	%eax
– Immediate data	idata	$0x1234
– Direct (memory)	mem	label (in source)
– Register Indirect	mem	(%eax)
– Register Indirect with fixed displacement	mem	disp(%eax)
– Offset (for jcc/jmp)	mem	label (in source)
– Port (for in/out)	idata (%dx)	$0x12 (%dx)

ADD

Integer Addition

Syntax:

addb src, dest
addw src, dest
addl src, dest

Operation:

dest ← dest + src

Legal Operands

src	dest
idata,	reg
idata,	mem
reg,	reg
mem,	reg
reg,	mem

Examples:

addl $10, %eax
addb $10, label
addw %bx, %ax
addl label, %eax
addl %eax, label

Description

This instruction adds the contents of the dest and src operands and stores the result in the location specified by dest. The operands must be of the same size. If the operands are signed integers, the OF flag indicates an invalid result. If the operands are unsigned, the CF flag indicates a carry out of the destination. If the operands are unpacked BCD digits, the AF flag indicates a decimal carry.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
x	-	-	-	x	x	-	x	-	x	-	x

AND

Boolean AND

Syntax:

andb src, dest
andw src, dest
andl src, dest

Operation:

dest ← dest & src

Legal Operands	
src	dest
idata,	reg
idata,	mem
reg,	reg
mem,	reg
reg,	mem

Examples:

andl $10, %eax
andb $10, label
andw %bx, %ax
andl label, %eax
andl %eax, label

Description

This instruction performs a bit by bit AND operation on the dest and src operands and stores the result in the dest operand. The AND operation is defined as:

AND	**0**	**1**
0	**0**	**0**
1	**0**	**1**

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
0	-	-	-	x	x	-	?	-	x	-	0

CALL

Near Procedure Call

Syntax:
call dest

Operation:
push %eip
%eip ← dest

Legal Operands		**Examples:**
dest		
offset	%eip ← %eip + offset	call label
mem	%eip ← contents of mem	call (%eax)
reg	%eip ← contents of reg	call %eax

Description

This instruction pushes the address of the next instruction (EIP) onto the stack. The instruction pointer is then set to the value of the operand.

If the operand is an offset, the operand value is a memory addres relative to the current value of %eip. If the operand is a memory address or a register, the subroutine address is taken indirectly from the operand.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
-	-	-	-	-	-	-	-	-	-	-	-

CLI

Clear Interrupt Enable Flag

Syntax:	**Legal Operands**	**Examples:**
cli	none	cli

Operation:

IF = 0

Description

This instruction clears the interrupt enable flag (IF) and disables the processing of interrupts. This instruction is used to prevent interrupts during short sequences of code that could fail if an interrupt were allowed to occur in the middle of the code sequence. The IF should not be turned off for "long" periods of time as this could prevent the processing of critical I/O operations such as causing incoming data to be overrun before the processor can execute the ISR code required to process it.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
-	-	0	-	-	-	-	-	-	-	-	-

CMP

Compare Integers

Syntax:

cmpb op1, op2
cmpw op1, op2
cmpl op1, op2

Operation:

NULL ← op2 – op1

Legal Operands

op1	op2
idata,	reg
idata,	mem
reg,	reg
mem,	reg
reg,	mem

Examples:

cmpl $10, %eax
cmpb $10, label
cmpw %bx, %ax
cmpl label, %eax
cmpl %eax, label

Description

This instruction subtracts the contents of the src operands from the dest operand and discards the result. Only the eflags register is affected as follows:

Condition	Signed Compare	Unsigned Compare
op1 < op2	ZF == 0 && SF == OF	CF == 0 && ZF == 0
op1 <= op2	SF == OF	CF == 0
op1 == op2	ZF == 1	ZF == 1
op1 >= op2	ZF ==1 || SF != OF	CF == 1 || ZF == 1
op1 > op2	SF != OF	CF ==1

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
x	-	-	-	x	x	-	x	-	x	-	x

DEC

Decrement

Syntax:	**Legal Operands**	**Examples:**
decb op1	op1	
decw op1	reg	decl %eax
decl op1	mem	decl label

Operation:

op1 ← op1 - 1

Description

This instruction subtracts the value 1 from op1. This instruction is often used to decrement indexes and therefore does not affect the carry flag (CF). In all other respects, it is equivalent to the instruction:

```
subb    $1, op1
```

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
x	-	-	-	x	x	-	x	-	x	-	-

IN

Input from I/O Port

Syntax:

inb port, %al
inw port, %ax
inl port, %eax

Legal Operands

port
idata (one byte)
(%dx)

Examples:

inw $0x72, %ax
inb (%dx), %al

Operation:

reg ← src (port)

Description

This instruction reads a byte, word, or long word into the specified accumulator from the designated I/O port. If you use an immediate data value in the instruction, you can address only the first 256 ports. If the port is specified in the %dx register, you can access any of the 65536 ports.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
-	-	-	-	-	-	-	-	-	-	-	-

INC

Increment

Syntax:	**Legal Operands**	**Examples:**
incb op1	op1	
incw op1	reg	incl %eax
incl op1	mem	incl label

Operation:

op1 ← op1 + 1

Description

This instruction adds the value 1 to op1. This instruction is often used to increment indexes and therefore does not affect the carry flag (CF). In other respects, it is equivalent to the instruction:

```
addb    $1, op1
```

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
x	-	-	-	x	x	-	x	-	x	-	-

INT

Software Interrupt

Syntax:

int vector

Legal Operands

vector

idata

Examples:

int $3

Operation:

push %eflags
push %cs
push %eip
TF ← 0
if (IDT(vector).type = INTERRUPT_GATE) IF ← 0
%eip ← destination (IDT(vector))

Description

This instruction is used as a system call. The int 3 instruction is usually encoded as a single byte 0xcc and used as a breakpoint instruction for debuggers.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
-	-	x	0	-	-	-	-	-	-	-	-

IRET

Interrupt Return

Syntax:

iret

Legal Operands

none

Examples:

iret

Operation:

(if check as option for task return is omitted here)
pop %eip
pop %cs
pop %eflags

Description

This instruction signals a return from an interrupt. NOTE: All of the pops shown are executed before the processor starts execution at the restored value of %eip. The three pops are handled as an "atomic" operation, i.e. executed as a single unit.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
x	x	x	x	x	x	-	x	-	x	-	x

Jcc

Jump if Condition

Syntax:	**Legal Operands**	**Examples:**
jcc offset	offset	
	mem	jne label

Operation:

if (cc) %eip ← %eip + sign_extend (offset)

Description

This instruction executes a conditional jump. It does not change the state of the flags. It executes the jump based on the value(s) of the flag bits as follows:

			After cmp_ x, y
ja	jump above	CF == 0 && ZF == 0	unsigned y > x
jae	jump above or equal	CF == 0	unsigned y >= x
jb	jump below	CF == 1	unsigned y < x
jbe	jump below or equal	CF == 1 || ZF == 1	unsigned y <= x
jc	jump if carry	CF == 1	
jcxz	jump if %cx == 0		
jecxz	jump if %ecx == 0		

Jcc (Continued)

Jump if Condition

Description (Continued)

				After cmp_ x, y
je	jump equal	ZF == 1		y == x
jg	jump greater	SF == OF && ZF = 0	signed	y > x
jge	jump greater or equal	SF == OF	signed	y >= x
jl	jump less	SF != OF	signed	y < x
jle	jump less or equal	SF != OF || ZF == 1	signed	y <= x
jna	jump not above	(same as jbe)		
jnae	jump not above or equal	(same as jb)		
jnb	jump not below	(same as jae)		
jnbe	jump not below or equal	(same as ja)		
jnc	jump no carry	CF == 0		
jne	jump not equal	ZF == 0		y != x
jng	jump not greater	(same as jle)		
jnge	jump not greater or equal	(same as jl)		
jnl	jump not less	(same as jge)		
jnle	jump not less or equal	(same as jg)		

Jcc (Continued)

Jump if Condition

Description (Continued)

jno	jump no overflow	OF == 0
jnp	jump no parity	PF == 0
jns	jump no sign	SF == 0
jnz	jump not zero	ZF == 0
jo	jump if overflow	OF == 1
jp	jump if parity	PF == 1
jpe	jump if parity even	PF == 1
jpo	jump if parity odd	PF == 0
js	jump if sign	SF == 1
jz	jump if zero	ZF == 1

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
-	-	-	-	-	-	-	-	-	-	-	-

JMP

Jump

Syntax:

jmp dest

Legal Operands		**Examples:**
dest		
offset	%eip ← %eip + offset	jmp label
mem	%eip ← contents of mem	jmp (%eax)
reg	%eip ← contents of reg	jmp %eax

Operation:

%eip ← dest

Description

This instruction executes an unconditional jump. It doesn't change the state of the flags.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
-	-	-	-	-	-	-	-	-	-	-	-

LEA

Load Effective Address

Syntax:

lea src, dest

Legal Operands

src	dest
mem,	reg

Examples:

lea label, %eax

Operation:

dest ← address (src)

Description

This instruction loads the address specified by the memory operand into the destination register. No memory access cycle takes place. It doesn't change the state of the flags.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
-	-	-	-	-	-	-	-	-	-	-	-

LOOPcc

Decrement %ecx and Branch

Syntax:	**Legal Operands**	**Examples:**
loop offset	Offset	
loopz offset	mem	loop label
loopnz offset		
loope offset		
loopne offset		

Operation:

%ecx ← %ecx - 1

if (cc & (%ecx != 0)) %eip ← %eip + offset

Description

These instructions support a decrement and branch operation. For all variants other than LOOP, the decrement and branch is combined with a test on the ZF bit. A loop counter is assumed in the register %ecx.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
-	-	-	-	-	-	-	-	-	-	-	-

MOV

Move Data

Syntax:

movb src, dest
movw src, dest
movl src, dest

Operation:

dest ← src

Legal Operands

src	dest
idata,	reg
idata,	mem
reg,	reg
mem,	reg
reg,	mem

Examples:

movl $10, %eax
movb $10, label
movw %bx, %ax
movl label, %eax
movl %eax, label

Description

This instruction copies the contents of the src operand into dest.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
-	-	-	-	-	-	-	-	-	-	-	-

NEG

Not

Syntax:	**Legal Operands**	**Examples:**
negb op1	op1	
negw op1	reg	negl %eax
negl op1	mem	negl label

Operation:

op1 ← - op1

Description

This instruction performs a two's complement on the operand.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
x	-	-	-	x	x	-	x	-	x	-	x

NOP

No Operation

Syntax:

nop

Legal Operands

none

Examples:

nop

Operation:

(nothing)

Description

This instruction does nothing except take time to be executed. Hence, it is used in timing loops or where the execution of the next instruction needs to be delayed for some reason, e.g. giving enough time for a hardware register to be ready.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
-	-	-	-	-	-	-	-	-	-	-	-

NOT

Not

Syntax:	**Legal Operands**	**Examples:**
notb op1	op1	
notw op1	reg	notl %eax
notl op1	mem	notl label

Operation:

op1 ← ~ op1

Description

This instruction performs a logical NOT or one's complement on the operand. The flags are unaffected.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
-	-	-	-	-	-	-	-	-	-	-	-

OR

Boolean OR

Syntax:

orb src, dest
orw src, dest
orl src, dest

Operation:

dest ← dest | src

Legal Operands	
src	dest
idata,	reg
idata,	mem
reg,	reg
mem,	reg
reg,	mem

Examples:

orl $10, %eax
orb $10, label
orw %bx, %ax
orl label, %eax
orl %eax, label

Description

This instruction performs a bit by bit OR operation on the dest and src operands and stores the result in the dest operand. The OR operation is defined as:

OR	**0**	**1**
0	**0**	**1**
1	**1**	**1**

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
0	-	-	-	x	x	-	?	-	x	-	0

OUT

Output to I/O Port

Syntax:

outb %al, port
outw %ax, port
outl %eax, port

Legal Operands

port
idata (one byte)
(%dx)

Examples:

outb %al, $0x72
outw %ax, (%dx)

Operation:

dest (port) ← reg

Description

This instruction writes a byte, word, or long word from the specified accumulator to the designated I/O port. If you use an immediate data value in the instruction, you can address only the first 256 ports. If the port is specified in the %dx register, you can access any of the 65536 ports.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
-	-	-	-	-	-	-	-	-	-	-	-

POP

Pop Value off Stack

Syntax:

popw dest

popl dest

Legal Operands

dest

reg

mem

Examples:

popl %eax

popw label

Operation:

dest ← contents of mem at %esp

%esp ← %esp + (w)? 2 : 4

Description

This instruction pops the current value at the top of the stack (lowest memory address), stores it in the dest operand, and increments the stack pointer by the size of the value.

popl is always preferred to keep the stack pointer aligned on long word boundaries, i.e. addresses with the two LSBs = 0.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
-	-	-	-	-	-	-	-	-	-	-	-

PUSH

Push Value onto Stack

Syntax:

pushw src
pushl src

Legal Operands

src
idata
reg
mem

Examples:

pushw $7
pushl %eax
pushw label

Operation:

%esp ← %esp - (w)? 2 : 4
contents of mem at %esp ← src

Description

This instruction decrements the stack pointer by the size of the value, and stores the value of the src operand onto the top of the stack (lowest memory address).

pushl is always preferred to keep the stack pointer aligned on long word boundaries, i.e. addresses with the two LSBs = 0.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
-	-	-	-	-	-	-	-	-	-	-	-

RET

Near Return from Subroutine

Syntax:

ret count

Legal Operands

count

(none)

idata

Examples:

ret

ret $4

Operation:

%eip ← pop (%esp)

%esp ← %esp + count

Description

This instruction restores the instruction pointer to the value it held before the previous call instruction. The value of the EIP that had been saved on the stack is popped. If the count operand is present, the count value is added to %esp, removing arguments that were pushed onto the stack for the subroutine call.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
-	-	-	-	-	-	-	-	-	-	-	-

SAL / SHL

Shift Arithmetic Left / Shift Logical Left

Syntax:

salb count, dest
salw count, dest
sall count, dest

Legal Operands

count	dest
idata	reg
idata	mem
%cl	reg
%cl	mem

Examples:

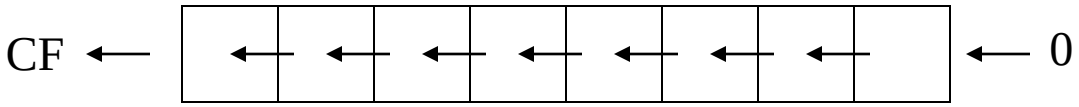
salw $4, %ax
salb $4, label
shll %cl, %eax
shlw %cl, label

Operation:

dest ← dest << count

Description

This instruction shifts the dest operand count bits to the left and fills the LSBs with zeros. It updates the flag bits appropriately. (Arithmetic and logical are the same.)



Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
x	-	-	-	x	x	-	-	-	x	-	x

SAR

Shift Arithmetic Right

Syntax:

sarb count, dest
sarw count, dest
sarl count, dest

Legal Operands

count	dest
idata	reg
idata	mem
%cl	reg
%cl	mem

Examples:

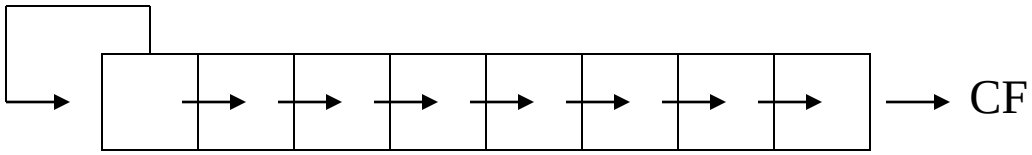
sarw $4, %ax
sarb $4, label
sarl %cl, %eax
sarw %cl, label

Operation:

dest ← dest >> count
(with sign bit extension)

Description

This instruction shifts the dest operand count bits to the right and fills the MSBs with copies of the sign bit. It updates the flag bits appropriately. (Preserves sign.)



Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
x	-	-	-	x	x	-	-	-	x	-	x

SHR

Shift Logical Right

Syntax:

shrb count, dest
shrw count, dest
shrl count, dest

Legal Operands

count	dest
idata	reg
idata	mem
%cl	reg
%cl	mem

Examples:

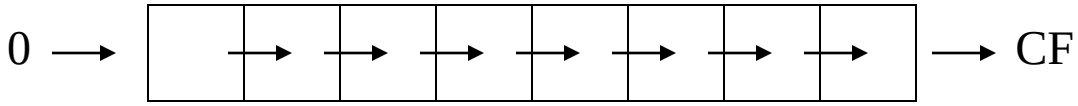
shrw $4, %ax
shrb $4, label
shrl %cl, %eax
shrw %cl, label

Operation:

dest ← dest >> count
(without sign bit extension)

Description

This instruction shifts the dest operand count bits to the right and fills the MSBs with zeros. It updates the flag bits appropriately. (Does not preserve sign.)



Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
x	-	-	-	x	x	-	-	-	x	-	x

STI

Set Interrupt Enable Flag

Syntax:	**Legal Operands**	**Examples:**
sti	none	sti

Operation:

IF = 1

Description

This instruction sets the interrupt enable flag (IF) and enables the processing of interrupts. This instruction is used when the code is ready to process interrupts.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
-	-	1	-	-	-	-	-	-	-	-	-

SUB

Integer Subtraction

Syntax:

subb src, dest
subw src, dest
subl src, dest

Operation:

dest ← dest - src

Legal Operands

src	dest
idata,	reg
idata,	mem
reg,	reg
mem,	reg
reg,	mem

Examples:

subl $10, %eax
subb $10, label
subw %bx, %ax
subl label, %eax
subl %eax, label

Description

This instruction subtracts the contents of the src operand from the dest operand and stores the result in the location specified by dest. The operands must be of the same size. If the operands are signed integers, the OF flag indicates an invalid result. If the operands are unsigned, the CF flag indicates a borrow into the destination. If the operands are unpacked BCD digits, the AF flag indicates a decimal borrow.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
x	-	-	-	x	x	-	x	-	x	-	x

TEST

Logical Compare

Syntax:

testb src, dest
testw src, dest
testl src, dest

Legal Operands

src	dest
idata,	reg
reg,	reg
mem,	reg

Examples:

testl $10, %eax
testw %bx, %ax
testl label, %eax

Operation:

NULL ← dest & src

Description

This instruction ANDs the contents of the src operand with the dest operand and discards the result. It sets the flags.

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
0	-	-	-	x	x	-	?	-	x	-	0

XOR

Boolean XOR

Syntax:

xorb src, dest
xorw src, dest
xorl src, dest

Operation:

dest ← dest ^ src

Legal Operands

src	dest
idata,	reg
idata,	mem
reg,	reg
mem,	reg
reg,	mem

Examples:

xorl $10, %eax
xorb $10, label
xorw %bx, %ax
xorl label, %eax
xorl %eax, label

Description

This instruction performs a bit by bit XOR operation on the dest and src operands and stores the result in the dest operand. The XOR operation is defined as:

XOR	**0**	**1**
0	**0**	**1**
1	**1**	**0**

Flags

OF	DF	IF	TF	SF	ZF		AF		PF		CF
0	-	-	-	x	x	-	?	-	x	-	0