Assembly Language Lecture Series:

x86-64 Flag Control Instructions

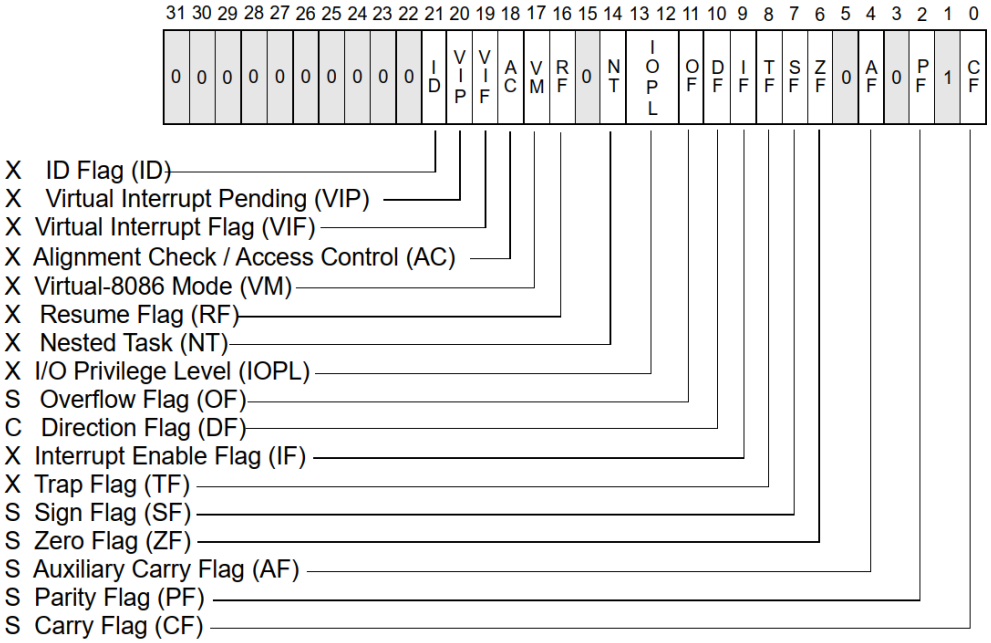
Sensei RL Uy, College of Computer Studies,
De La Salle University, Manila, Philippines

Copyright Notice

x86-64 Flag Control Instructions

1. **LAHF**
   load status flags into register AH
2. **SAHF**
   store AH into flags
3. **PUSHFQ**
   Push RFLAGS onto the stack
4. **POPFQ**
   pop flags into RFLAGS register
5. **CLC**
   Clear Carry Flag
6. **STC**
   Set Carry Flag
7. **CMC**
   Complement Carry Flag

Lower 32-bit RFLAGS register



The upper 32 bits of RFLAGS register is reserved.

x86-64 Flag Control Instructions: SAHF

SAHF (store AH into flags)

Syntax: SAHF

SF	ZF	0	AF	0	PF	1	CF	←	AH

Flags affected:

SF, ZF, AF, PF, CF

x86-64 Flag Control Instructions: SAHF

SAHF (store AH into flags)

Syntax: SAHF

SF	ZF	0	AF	0	PF	1	CF	←	AH

Flags affected:
SF, ZF, AF, PF, CF

Example:

```
section .text
MOV AH, 0xFF
SAHF
```

1. **What will SF, ZF, AF, PF, CF contain after execution?**

x86-64 Flag Control Instructions: SAHF

SAHF (store AH into flags)

Syntax: SAHF

SF	ZF	0	AF	0	PF	1	CF	←	AH

Flags affected:
SF, ZF, AF, PF, CF

Example:

```
section .text
MOV AH, 0xFF
SAHF
```

1. What will SF, ZF, AF, PF, CF contain after execution?

SF = 1
ZF = 1
AF = 1
PF = 1
CF = 1

x86-64 Flag Control Instructions: POPFQ

POPFQ (pop flags into RFLAGS register)

Syntax: POPFQ

RFLAGS ← [RSP]

RSP ← RSP+8

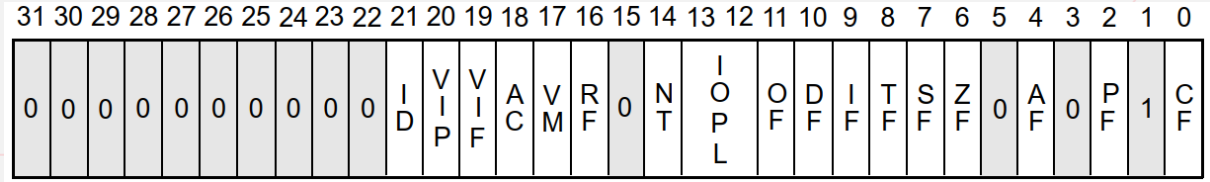
Flags affected:

*all status flags are affected

*all non-reserved bits (except IOPL,VIP,VIF,VM and RF) can be modified.

*IOPL, VIP, VIF, VM, and all reserved bits are unaffected

*RF=0



x86-64 Flag Control Instructions: POPFQ

POPFQ (pop flags into RFLAGS register)

Syntax: POPFQ
RFLAGS ← [RSP]
RSP ← RSP+8

Flags affected:
*all status flags are affected
*all non-reserved bits (except IOPL,VIP,VIF,VM and RF) can be modified.
*IOPL, VIP, VIF, VM, and all reserved bits are unaffected
*RF=0

Example:

```
section .text
MOV RAX, 0xFFFF_FFFF_FFFF_FFFF
PUSH RAX
POPFQ
```

1. **What will the flags contain after execution?**

x86-64 Flag Control Instructions: POPFQ

POPFQ (pop flags into RFLAGS register)

Syntax: POPFQ
RFLAGS ← [RSP]
RSP ← RSP+8

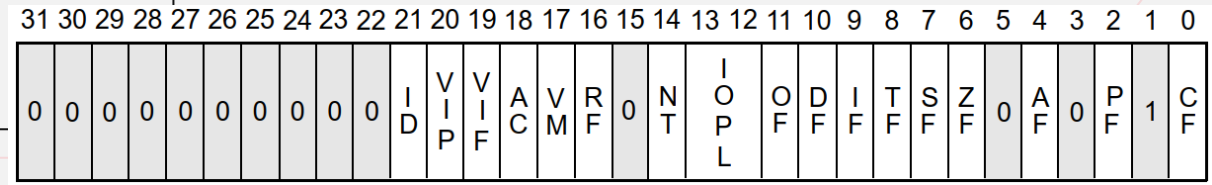
Flags affected:
*all status flags are affected
*all non-reserved bits (except IOPL,VIP,VIF,VM and RF) can be modified.
*IOPL, VIP, VIF, VM, and all reserved bits are unaffected
*RF=0

Example:

```
section .text
MOV RAX, 0xFFFF_FFFF_FFFF_FFFF
PUSH RAX
POPFQ
```

1. What will the flags contain after execution?

The following flags are set to 1:
1. Status flags: CF, PF, AF, ZF, SF, OF
2. Control flag: DF
3. System flags: IF, NT, AC, ID



x86-64 Flag Control Instructions: LAHF

LAHF (load status flags into register AH)

Syntax: LAHF

AH	←	SF	ZF	0	AF	0	PF	1	CF

Flags affected:

None

x86-64 Flag Control Instructions: LAHF

LAHF (load status flags into register AH)

Syntax: LAHF

AH	←	SF	ZF	0	AF	0	PF	1	CF

Flags affected:
None

Example:

```
section .text
MOV RAX,
0xFFFF_FFFF_FFFF_FFFF
PUSH RAX
POPFQ
LAHF
```

1. **What will the AH contain after execution?**

x86-64 Flag Control Instructions: LAHF

LAHF (load status flags into register AH)

Syntax: LAHF

AH	←	SF	ZF	0	AF	0	PF	1	CF

Flags affected:
None

Example:

```
section .text
MOV RAX,
0xFFFF_FFFF_FFFF_FFFF
PUSH RAX
POPFQ
LAHF
```

1. What will the AH contain after execution?

   AH = D7

x86-64 Flag Control Instructions: PUSHFQ

PUSHFQ (push RFLAGS onto the stack)

Syntax: PUSHFQ

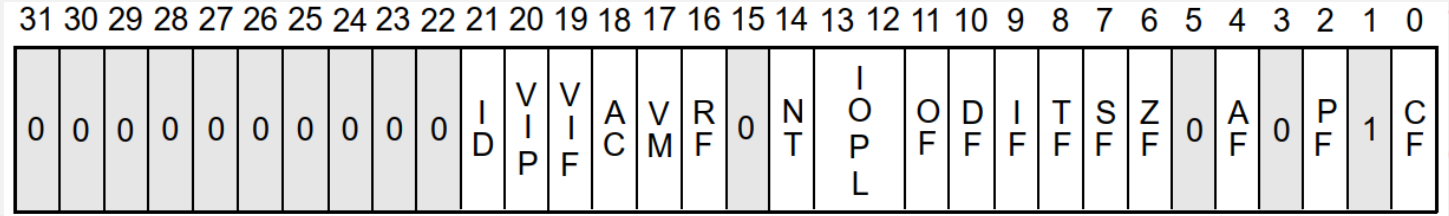
RSP ← RSP-8

[RSP] ← RFLAGS

Note: The VM and RF flags (bits 16 and 17) are not copied; instead, values for these flags are cleared in the RFLAGS image stored on the stack.

Flags affected:

None



x86-64 Flag Control Instructions: PUSHFQ

PUSHFQ (push RFLAGS onto the stack)

Syntax: PUSHFQ

RSP ← RSP-8
[RSP] ← RFLAGS

Note: The VM and RF flags (bits 16 and 17) are not copied; instead, values for these flags are cleared in the RFLAGS image stored on the stack.

Flags affected:

None

Example:

```
section .text
SUB RCX,RCX
PUSHFQ
POP RAX
```

1. **What will RAX contain after execution?**

x86-64 Flag Control Instructions: PUSHFQ

PUSHFQ (push RFLAGS onto the stack)

Syntax: PUSHFQ

RSP ← RSP-8
[RSP] ← RFLAGS

Note: The VM and RF flags (bits 16 and 17) are not copied; instead, values for these flags are cleared in the RFLAGS image stored on the stack.

Flags affected:

None

Example:

```
section .text
SUB RCX,RCX
PUSHFQ
POP RAX
```

1. What will RAX contain after execution?

RAX = 0000_0000_0000_0346

*least significant 32 bits(in binary):
0000_0000_0000_0000_0000_0011_0100_0110

x86-64 Flag Control Instructions: CLC/STC/CMC

CLC (clear carry flag)

Syntax: CLC
CF = 0
Flags affected:
*CF
*OF, ZF, SF, AF, PF: no change

CMC (complement carry flag)

Syntax: CMC
CF = CF’
Flags affected:
*CF
*OF, ZF, SF, AF, PF: no change

STC (set carry flag)

Syntax: STC
CF = 1
Flags affected:
*CF
*OF, ZF, SF, AF, PF: no change

x86-64 Flag Control Instructions: CLC/STC/CMC

CLC (clear carry flag)

Syntax: CLC
CF = 0
Flags affected:
*CF
*OF, ZF, SF, AF, PF: no change

STC (set carry flag)

Syntax: STC
CF = 1
Flags affected:
*CF
*OF, ZF, SF, AF, PF: no change

CMC (complement carry flag)

Syntax: CMC
CF = CF'
Flags affected:
*CF
*OF, ZF, SF, AF, PF: no change

Example:

```
section .text
SUB RCX,RCX
CMC
```

1. What will carry flag (CF) contain after execution?

x86-64 Flag Control Instructions: CLC/STC/CMC

CLC (clear carry flag)

Syntax: CLC
CF = 0
Flags affected:
*CF
*OF, ZF, SF, AF, PF: no change

STC (set carry flag)

Syntax: STC
CF = 1
Flags affected:
*CF
*OF, ZF, SF, AF, PF: no change

CMC (complement carry flag)

Syntax: CMC
CF = CF'
Flags affected:
*CF
*OF, ZF, SF, AF, PF: no change

Example:

```
section .text
SUB RCX,RCX
CMC
```

1. What will carry flag (CF) contain after execution?

CF = 1