x86-64 Assembly Language Instruction Summary

Summarized by Soraya Abad-Mota

Spring 2017

This document presents a summary of all the instructions for the assembly language of the x86-64 processor, presented in the textbook by Bryant and O'Hallaron: "Computer Systems: A programmer's perspective", 3rd Edition.

The document is organized into three sections and five subsections. Each section or subsection summarizes a classes of instructions in tables that reproduces the information in the respective figures from the textbook. For example Table 4 contains the integer arithmetic operations from figure 3.10. All the tables have the same format: the first column shows the name of the operation with its operands, the second column shows the effect of the operation on the operands or the condition applied, and the third column has a brief description of the operation.

1 Data Movement instructions

The first instructions covered are the ones that move data between locations.

1.1 Simple move Instructions

Figure 3.4 on page 183 contains the four (4) simple moves depending on the size of the operands plus a special operation called `movabsq`. The `movabsq` special instruction can have an arbitrary

Table 1: Simple Move Instructions

Instruction	Effect	Description
movb S,D	$D \leftarrow S$	Move byte
movw S,D	$D \leftarrow S$	Move word
movl S,D	$D \leftarrow S$	Move double word
movq S,D	$D \leftarrow S$	Move quad word
movabsq I,R	$R \leftarrow I$	Move absolute quad word

64-bit immediate value as its source operand and can only have a register as a destination.

1.2 Zero extending data movement instructions

Figure 3.5 on page 184 contains five (5) move instructions that fill out the destination higher bytes with zeros as indicated by the operation suffix (bw, bl, wl, bq, or wq). These instructions have a register or a memory location as the source and a register as the destination.

Table 2: Zero Extending Move Instructions

Instruction	Effect	Description
movzbw S,R	$R \leftarrow ZeroExtend(S)$	Move zero-extended byte to word
movzbl S,R	$R \leftarrow ZeroExtend(S)$	Move zero-extended byte to double word
movzwl S,R	$R \leftarrow ZeroExtend(S)$	Move zero-extended word to double word
movzbq S,R	$R \leftarrow ZeroExtend(S)$	Move zero-extended byte to quad word
movzwq S,R	$R \leftarrow ZeroExtend(S)$	Move zero-extended word to quad word

1.3 Sign extending data movement instructions

Figure 3.6 on page 185 shows six (6) sign-extension move instructions and the special instruction called `cltq`, with no operands always uses those two registers to move the contents of one to the other with sign-extension.

Table 3: Sign-Extending Move Instructions

Instruction	Effect	Description
movsbw S,R	$R \leftarrow SignExtend(S)$	Move sign-extended byte to word
movsbl S,R	$R \leftarrow SignExtend(S)$	Move sign-extended byte to double word
movswl S,R	$R \leftarrow SignExtend(S)$	Move sign-extended word to double word
movsbq S,R	$R \leftarrow SignExtend(S)$	Move sign-extended byte to quad word
movswq S,R	$R \leftarrow SignExtend(S)$	Move sign-extended word to quad word
movslq S,R	$R \leftarrow SignExtend(S)$	Move sign-extended double word to quad word
cltq	%rax $\leftarrow SignExtend(\%eax)$	Sign-extend %eax to %rax

2 Integer Arithmetic Instructions

Figure 3.10 on page 192 displays the 14 integer arithmetic operations and the special `leaq` operation to find the effective address.

Table 4: Integer Arithmetic Instructions

Instruction	Effect	Description
leaq S, D	$D \leftarrow \&S$	Load Effective address. Copy the address of S to D
$INC\ \ D$	$D \leftarrow D + 1$	Increment
$DEC\ \ D$	$D \leftarrow D - 1$	Decrement
$NEG\ \ D$	$D \leftarrow -D$	Negate
$NOT\ \ D$	$D \leftarrow \neg D$	Complement
ADD S,D	$D \leftarrow D + S$	Add
SUB S,D	$D \leftarrow D - S$	Subtract
IMUL S,D	$D \leftarrow D * S$	Multiply
XOR S,D	$D \leftarrow D \wedge S$	Exclusive-or
OR S,D	$D \leftarrow D \mid S$	Or
AND S,D	$D \leftarrow D\&S$	And
SAL k,D		Left shift
SHL k,D		Left shift (same as SAL)
SAR k,D		Arithmetic right shift
SHR k,D		Logical right shift

In addition, there are some special arithmetic operations for full 128-bit multiplication and division, which use registers %rdx and %rax as a single 128-bit word. These operations are presented in figure 3.12 on page 198, you may look for them in the textbook and are not summarized in this document.

3 Instructions to set and use the Condition Codes

The arithmetic operations implicitly set the condition code 1-bit registers. The eight (8) operations on Figure 3.13 on page 202 explicitly set the 4 condition code registers doing compare and test.

Table 5: Comparison and Test Instructions

Instruction	Based on	Description
cmpb S_1, S_2	$S_2 - S_1$	Compare byte
cmpw S_1, S_2	$S_2 - S_1$	Compare word
cmpl S_1, S_2	$S_2 - S_1$	Compare double word
cmpq S_1, S_2	$S_2 - S_1$	Compare quad word
testb S_1, S_2	$S_2 \& S_1$	Test byte
testw S_1, S_2	$S_2 \& S_1$	Test word
testl S_1, S_2	$S_2 \& S_1$	Test double word
testq S_1, S_2	$S_2 \& S_1$	Test quad word

3.1 Set Instructions

Figure 3.14 on page 203, show the 12 set instructions which use the condition codes to set the low-order byte of their destination.

Table 6: The Set Instructions

Instruction	Effect	Set Condition
sete D	$D \leftarrow ZF$	Equal/zero
setne D	$D \leftarrow \neg ZF$	Not equal/not zero
sets D	$D \leftarrow SF$	Negative
setns D	$D \leftarrow \neg SF$	Nonnegative
setg D	$D \leftarrow \neg(SF \wedge OF)$ & $\neg ZF$	Greater (signed $>$)
setge D	$D \leftarrow \neg(SF \wedge OF)$	Greater or equal (signed $\geq$)
setl D	$D \leftarrow (SF \wedge OF)$	Less (signed $<$)
setle D	$D \leftarrow (SF \wedge OF) \mid ZF$	Less or equal (signed $\leq$)
seta D	$D \leftarrow \neg CF$ & $\neg ZF$	Above (unsigned $>$)
setae D	$D \leftarrow \neg CF$	Above or equal (unsigned $\geq$)
setb D	$D \leftarrow CF$	Below (unsigned $<$)
setbe D	$D \leftarrow CF \mid ZF$	Below or equal (unsigned $\leq$)

3.2 Jump and Conditional Move Instructions

Figure 3.15 on page 206 presents the Jump Instructions, which match the set instructions.

Table 7: The Jump Instructions

Instruction	Jump Condition	Description
jmp *Label*	1	Direct (unconditional) jump
jmp $*operand$	1	Indirect (unconditional) jump
je *Label*	ZF	Equal/zero
jne *Label*	$\neg ZF$	Not Equal/ not zero
js *Label*	SF	Negative
jns *Label*	$\neg SF$	Nonnegative
jg *Label*	$\neg(SF \wedge OF)$ & $\neg ZF$	Greater (signed $>$)
jge *Label*	$\neg(SF \wedge OF)$	Greater or equal (signed $\geq$)
jl *Label*	$(SF \wedge OF)$	Less (signed $<$)
jle *Label*	$(SF \wedge OF) \mid ZF$	Less or equal (signed $\leq$)
ja *Label*	$\neg CF$ & $\neg ZF$	Above (unsigned $>$)
jae *Label*	$\neg CF$	Above or equal (unsigned $\geq$)
jb *Label*	CF	Below (unsigned $<$)
jbe *Label*	$CF \mid ZF$	Below or equal (unsigned $\leq$)

Figure 3.18 on page 217 show the 12 conditional move instructions, shown here on Table 8.

Table 8: The Conditional Move Instructions

Instruction	Move Condition	Description
cmove S,R	ZF	Equal/zero
cmovne S,R	$\neg ZF$	Not Equal/ not zero
cmovs S,R	SF	Negative
cmovns S,R	$\neg SF$	Nonnegative
cmovg S,R	$\neg(SF \wedge OF)$ & $\neg ZF$	Greater (signed $>$)
cmovge S,R	$\neg(SF \wedge OF)$	Greater or equal (signed $\geq$)
cmovl S,R	$(SF \wedge OF)$	Less (signed $<$)
cmovle S,R	$(SF \wedge OF) \mid ZF$	Less or equal (signed $\leq$)
cmova S,R	$\neg CF$ & $\neg ZF$	Above (unsigned $>$)
cmovae S,R	$\neg CF$	Above or equal (unsigned $\geq$)
cmovb S,R	CF	Below (unsigned $<$)
cmovbe S,R	$CF \mid ZF$	Below or equal (unsigned $\leq$)