

CSCE 313

Embedded Systems Programming

2003/2/10

Spring 2003 – Lecture 11

M68000 Instruction Set

© 2003 Ms. Padmaja Sunkara – Lecturer
Some figures: MacKenzie © 1995 Prentice Hall Publishers, Inc.

68000 Instruction Set

68000 implements rich assortment of operations which can be categorized as:

- Integer Arithmetic
- Data movement
- Boolean
- Shift and rotate
- Bit manipulation
- Binary-coded decimal
- Program flow
- System control

68000 Instruction set

- To understand the operation of each instruction, the way condition code bits are affected is vital.

SR 271F

CCR

CCR 00011111

X N Z V C

X-EXTEND
Z-ZERO
C-CARRY

N-NEGATIVE
V-OVERFLOW



© 2002 Dr. James P. Davis Page 3

6800 Data Movement Instructions

- Data movement instructions provide facility to move data among memory locations or registers

Instruction	Operation
EXG	Exchange registers
LEA	Load effective address
LINK	Link and allocate stack
MOVE	Move source to destination
MOVEA	Move source to address register
MOVEM	Move multiple registers
MOVEP	Move to peripheral
MOVEQ	Move short data to destination
PEA	Push effective address
UNLK	Unlink stack

Source: MacKenzie © 1995 Prentice Hall Publishers



© 2002 Dr. James P. Davis Page 4

6800 Data Movement Instructions

- EXG-exchange contents of two registers
- LEA 6(A0,D5.L),A0-loads A0 with effective address
- PEA –moves EA to active stack rather than to address register.
- MOVEM.W D3-D5/A2, -(A7) –moves multiple registers to or from memory locations.

D3—W1	N-
D4—W2	N-2-----W4
D5—W3	N-4 -----W3
A2—W4	N-6 -----W2
A7—N	N-8 -----W1

- MOVEP-facilitates word or long word transfers to 8-bit peripherals.



68000 Instruction set

- LINK –has two purposes.To create a pointer to a parameter list and to create a temporary storage for local variables.
- UNLK deallocates storage occupied by local variables and restores old value of pointer

Refer the Figure 3.4(page 82) from Mackenzie



68000 Boolean Instructions

Source: MacKenzie © 1995 Prentice Hall Publishers

Instruction	Operation
AND	AND source to destination
ANDI	AND immediate data to destination
EOR	Exclusive OR source to destination
EORI	Exclusive OR immediate data to destination
NOT	Complement destination
OR	OR source to destination
ORI	OR immediate data to destination
Scc	Test condition codes and set operand
TST	Test operand and set condition codes



© 2002 Dr. James P. Davis Page 7

68000 Shift and Rotate Instructions

Source: MacKenzie © 1995 Prentice Hall Publishers

Instruction	Operation	Bit Movement
ASL	Arithmetic shift left	
ASR	Arithmetic shift right	
LSL	Logical shift left	
LSR	Logical shift right	
ROL	Rotate left	
ROR	Rotate right	
ROXL	Rotate left with extend bit	
ROXR	Rotate right with extend bit	
SWAP	Swap words of a longword	



© 2002 Dr. James P. Davis Page 8

68000 Instruction Set

- Arithmetic shift instructions treat data as signed integers. Shifting is analogous to multiplication by power of two (ASL) or division by power of two (ASR).
- Logical Shift instructions treat data as Boolean variables. One or more bits are lost depending on the shift count. If shift is left bits are lost from the most significant side of the operand. If the shift is right, bits are lost from least significant side.

ASR.B D3,D2

(SHIFT COUNT IS IN D3 AND SHIFT DATA IN D2)

- SWAP instruction is also available to exchange the upper and lower words in a 32 bit operand.



68000 Integer Arithmetic Instructions

Source: MacKenzie © 1995 Prentice Hall Publishers

Instruction	Operation
ADD	Add source to destination
ADDA	Add source to address register
ADDI	Add immediate data to destination
ADDQ	Add short data to destination
ADDX	Add with extend bit to destination
CLR	Clear operand
CMP	Compare source to destination
CMPA	Compare source to address register
CMPM	Compare memory
DIVS	Signed divide
DIVU	Unsigned divide
EXT	Sign extend
EXTB	Sign extend byte
MULS	Signed multiply
MULU	Unsigned multiply
NEG	Negate
NEGX	Negate with extend
SUB	Subtract source from destination
SUBA	Subtract source from address register
SUBI	Subtract immediate from destination
SUBQ	Subtract short from destination
SUBX	Subtract with extend bit from destination



68000 Integer Arithmetic Instructions

ADDX, SUBX permit multi-precision addition and subtraction operations by adding or subtracting the X bit as well as the two operands. It is supposed that X bit holds the carry from a previous operation.

ADD.L D1,D3

ADDX.L D0,D2

Ex for CMP: Refer Figure 3.5(page 83)



© 2002 Dr. James P. Davis Page 11

68000 Instruction Set

- DIVS instruction works with a 32-bit dividend and a 16-bit divisor(the source operand)(Ex:Figure 3.8(page 84)
- DIVS (unsigned divide) is same as DIVS except that both dividend and divisor are treated as unsigned integers.
- MULL multiplies two 16 bit unsigned integers and generates 32 bit product which are saved in destination register.
- CLR writes zeros into destination operand.
- NEG performs a 2's compliment operation on destination data
- EXT is used to increase bit size of signed integer.

EXT.W D5



© 2002 Dr. James P. Davis Page 12

68000 PROGRAM Flow Instructions

Instruction	Operation
• Bcc	Branch conditionally
• BRA	Branch Always
• BSR	Branch to subroutine
• DBcc	Test decrement and branch
• JMP	Jump to address
• JSR	Jump to subroutine
• NOP	No operation
• RTE	Return and reallocate stack
• RTR	Return and restore condition codes
• RTS	return from subroutine



68000 Instructions

✿ 68000 Bit manipulation Instructions

Instruction	Operand
BCHG	Change bit
BCLR	Clear bit
BSET	Set bit
BTST	Test bit EX:BTST #7,d5(bit 7 in D5)

✿ 68000 Binary coded decimal Instructions

Instruction	Operand
ABCD	add source to destination
NBCD	negate destination
SBCD	subtract source from destination



68000 System Control Instructions

Source: MacKenzie © 1995 Prentice Hall Publishers

Instruction	Operation
ANDI ^{tt}	AND immediate to status register/condition code register
CHK	Trap on upper out-of-bounds operand
EORI ^{tt}	Exclusive OR immediate to status register/condition code register
ILLEGAL	Illegal instruction trap
MOVE ^{tt}	Move to/from status register/condition code register
ORI ^{tt}	OR immediate to status register/condition code register
RESET ^t	Assert RESET line
STOP ^t	Stop processor
TAS	Test and set operand
TRAP	Trap unconditionally
TRAPV	Trap on overflow