HCS08 CPU
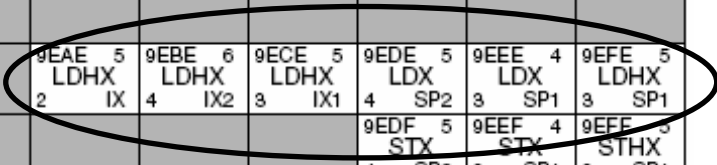
INSTRUCTION SET

Tsinghua Freescale MCU/DSP Application Center

HCS08 CPU指令表

Bit-Manipulation		Branch	Read-Modify-Write					Control		Register/Memory					
00 5 BRSET0 3 DIR	10 5 BSET0 2 DIR	20 3 BRA 2 REL	30 5 NEG 2 DIR	40 1 NEGA 1 INH	50 1 NEGX 1 INH	60 5 NEG 2 IX1	70 4 NEG 1 IX	80 9 RTI 1 INH	90 3 BGE 2 REL	A0 2 SUB 2 IMM	B0 3 SUB 2 DIR	C0 4 SUB 3 EXT	D0 4 SUB 3 IX2	E0 3 SUB 2 IX1	F0 3 SUB 1 IX
01 5 BRCLR0 3 DIR	11 5 BCLR0 2 DIR	21 3 BRN 2 REL	31 5 CBEQ 3 DIR	41 4 CBEQA 3 IMM	51 4 CBEQX 3 IMM	61 5 CBEQ 3 IX1+	71 5 CBEQ 2 IX+	81 6 RTS 1 INH	91 3 BLT 2 REL	A1 2 CMP 2 IMM	B1 3 CMP 2 DIR	C1 4 CMP 3 EXT	D1 4 CMP 3 IX2	E1 3 CMP 2 IX1	F1 3 CMP 1 IX
02 5 BRSET1 3 DIR	12 5 BSET1 2 DIR	22 3 BHI 2 REL	(32 5 LDHX EXT)	42 5 MUL 1 INH	52 6 DIV 1 INH	62 1 NSA 1 INH	72 1 DAA 1 INH	(82 BGND INH)	92 3 BGT 2 REL	A2 2 SBC 2 IMM	B2 3 SBC 2 DIR	C2 4 SBC 3 EXT	D2 4 SBC 3 IX2	E2 3 SBC 2 IX1	F2 3 SBC 1 IX
03 5 BRCLR1 3 DIR	13 5 BCLR1 2 DIR	23 3 BLS 2 REL	33 5 COM 2 DIR	43 1 COMA 1 INH	53 1 COMX 1 INH	63 5 COM 2 IX1	73 4 COM 1 IX	83 11 SWI 1 INH	93 3 BLE 2 REL	A3 2 CPX 2 IMM	B3 3 CPX 2 DIR	C3 4 CPX 3 EXT	D3 4 CPX 3 IX2	E3 3 CPX 2 IX1	F3 3 CPX 1 IX
04 5 BRSET2 3 DIR	14 5 BSET2 2 DIR	24 3 BCC 2 REL	34 5 LSR 2 DIR	44 1 LSRA 1 INH	54 1 LSRX 1 INH	64 5 LSR 2 IX1	74 4 LSR 1 IX	84 1 TAP 1 INH	94 2 TXS 1 INH	A4 2 AND 2 IMM	B4 3 AND 2 DIR	C4 4 AND 3 EXT	D4 4 AND 3 IX2	E4 3 AND 2 IX1	F4 3 AND 1 IX
05 5 BRCLR2 3 DIR	15 5 BCLR2 2 DIR	25 3 BCS 2 REL	35 4 STHX 2 DIR	45 3 LDHX 3 IMM	55 4 LDHX 2 DIR	65 3 CPHX 3 IMM	75 3 CPHX 2 DIR	85 1 TPA 1 INH	95 2 TSX 1 INH	A5 2 BIT 2 IMM	B5 3 BIT 2 DIR	C5 4 BIT 3 EXT	D5 4 BIT 3 IX2	E5 3 BIT 2 IX1	F5 3 BIT 1 IX
06 5 BRSET3 3 DIR	16 5 BSET3 2 DIR	26 3 BNE 2 REL	36 5 ROR 2 DIR	46 1 RORA 1 INH	56 1 RORX 1 INH	66 5 ROR 2 IX1	76 4 ROR 1 IX	86 3 PULA 1 INH	96 5 STHX 3 EXT	A6 2 LDA 2 IMM	B6 3 LDA 2 DIR	C6 4 LDA 3 EXT	D6 4 LDA 3 IX2	E6 3 LDA 2 IX1	F6 3 LDA 1 IX
07 5 BRCLR3 3 DIR	17 5 BCLR3 2 DIR	27 3 BEQ 2 REL	37 5 ASR 2 DIR	47 1 ASRA 1 INH	57 1 ASRX 1 INH	67 5 ASR 2 IX1	77 4 ASR 1 IX	87 2 PSHA 1 INH	97 1 TAX 1 INH	A7 2 AIS 2 IMM	B7 3 STA 2 DIR	C7 4 STA 3 EXT	D7 4 STA 3 IX2	E7 3 STA 2 IX1	F7 2 STA 1 IX
08 5 BRSET4 3 DIR	18 5 BSET4 2 DIR	28 3 BHCC 2 REL	38 5 LSL 2 DIR	48 1 LSLA 1 INH	58 1 LSLX 1 INH	68 5 LSL 2 IX1	78 4 LSL 1 IX	88 3 PULX 1 INH	98 1 CLC 1 INH	A8 2 EOR 2 IMM	B8 3 EOR 2 DIR	C8 4 EOR 3 EXT	D8 4 EOR 3 IX2	E8 3 EOR 2 IX1	F8 3 EOR 1 IX
09 5 BRCLR4 3 DIR	19 5 BCLR4 2 DIR	29 3 BHCS 2 REL	39 5 ROL 2 DIR	49 1 ROLA 1 INH	59 1 ROLX 1 INH	69 5 ROL 2 IX1	79 4 ROL 1 IX	89 2 PSHX 1 INH	99 1 SEC 1 INH	A9 2 ADC 2 IMM	B9 3 ADC 2 DIR	C9 4 ADC 3 EXT	D9 4 ADC 3 IX2	E9 3 ADC 2 IX1	F9 3 ADC 1 IX
0A 5 BRSET5 3 DIR	1A 5 BSET5 2 DIR	2A 3 BPL 2 REL	3A 5 DEC 2 DIR	4A 1 DECA 1 INH	5A 1 DECX 1 INH	6A 5 DEC 2 IX1	7A 4 DEC 1 IX	8A 3 PULH 1 INH	9A 1 CLI 1 INH	AA 2 ORA 2 IMM	BA 3 ORA 2 DIR	CA 4 ORA 3 EXT	DA 4 ORA 3 IX2	EA 3 ORA 2 IX1	FA 3 ORA 1 IX
0B 5 BRCLR5 3 DIR	1B 5 BCLR5 2 DIR	2B 3 BMI 2 REL	3B 7 DBNZ 3 DIR	4B 4 DBNZA 2 INH	5B 4 DBNZX 2 INH	6B 7 DBNZ 3 IX1	7B 6 DBNZ 2 IX	8B 2 PSHH 1 INH	9B 1 SEI 1 INH	AB 2 ADD 2 IMM	BB 3 ADD 2 DIR	CB 4 ADD 3 EXT	DB 4 ADD 3 IX2	EB 3 ADD 2 IX1	FB 3 ADD 1 IX
0C 5 BRSET6 3 DIR	1C 5 BSET6 2 DIR	2C 3 BMC 2 REL	3C 5 INC 2 DIR	4C 1 INCA 1 INH	5C 1 INCX 1 INH	6C 5 INC 2 IX1	7C 4 INC 1 IX	8C 1 CLRH 1 INH	9C 1 RSP 1 INH		BC 3 JMP 2 DIR	CC 4 JMP 3 EXT	DC 4 JMP 3 IX2	EC 3 JMP 2 IX1	FC 3 JMP 1 IX
0D 5 BRCLR6 3 DIR	1D 5 BCLR6 2 DIR	2D 3 BMS 2 REL	3D 4 TST 2 DIR	4D 1 TSTA 1 INH	5D 1 TSTX 1 INH	6D 4 TST 2 IX1	7D 3 TST 1 IX		9D 1 NOP 1 INH	AD 5 BSR 2 REL	BD 5 JSR 2 DIR	CD 6 JSR 3 EXT	DD 6 JSR 3 IX2	ED 5 JSR 2 IX1	FD 5 JSR 1 IX
0E 5 BRSET7 3 DIR	1E 5 BSET7 2 DIR	2E 3 BIL 2 REL	3E 6 CPHX 3 EXT	4E 5 MOV 3 DD	5E 5 MOV 2 DIX+	6E 4 MOV 3 IMD	7E 5 MOV 2 IX+D	8E 2+ STOP 1 INH	9E Page 2	AE 2 LDX 2 IMM	BE 3 LDX 2 DIR	CE 4 LDX 3 EXT	DE 4 LDX 3 IX2	EE 3 LDX 2 IX1	FE 3 LDX 1 IX
0F 5 BRCLR7 3 DIR	1F 5 BCLR7 2 DIR	2F 3 BIH 2 REL	3F 5 CLR 2 DIR	4F 1 CLRA 1 INH	5F 1 CLRX 1 INH	6F 5 CLR 2 IX1	7F 4 CLR 1 IX	8F 2+ WAIT 1 INH	9F 1 TXA 1 INH	AF 2 AIX 2 IMM	BF 3 STX 2 DIR	CF 4 STX 3 EXT	DF 4 STX 3 IX2	EF 3 STX 2 IX1	FF 2 STX 1 IX

Bit-Manipulation		Branch	Read-Modify-Write				Control				Register/Memory		
			9E60 6 NEG 3 SP1								9ED0 5 SUB 4 SP2	9EE0 4 SUB 3 SP1	
			9E61 6 CBEQ 4 SP1								9ED1 5 CMP 4 SP2	9EE1 4 CMP 3 SP1	
											9ED2 5 SBC 4 SP2	9EE2 4 SBC 3 SP1	
			9E63 6 COM 3 SP1								9ED3 5 CPX 4 SP2	9EE3 4 CPX 3 SP1	9EF3 6 CPHX 3 SP1
			9E64 6 LSR 3 SP1								9ED4 5 AND 4 SP2	9EE4 4 AND 3 SP1	
											9ED5 5 BIT 4 SP2	9EE5 4 BIT 3 SP1	
			9E66 6 ROR 3 SP1								9ED6 5 LDA 4 SP2	9EE6 4 LDA 3 SP1	
			9E67 6 ASR 3 SP1								9ED7 5 STA 4 SP2	9EE7 4 STA 3 SP1	
			9E68 6 LSL 3 SP1								9ED8 5 EOR 4 SP2	9EE8 4 EOR 3 SP1	
			9E69 6 ROL 3 SP1								9ED9 5 ADC 4 SP2	9EE9 4 ADC 3 SP1	
			9E6A 6 DEC 3 SP1								9EDA 5 ORA 4 SP2	9EEA 4 ORA 3 SP1	
			9E6B 8 DBNZ 4 SP1								9EDB 5 ADD 4 SP2	9EEB 4 ADD 3 SP1	
			9E6C 6 INC 3 SP1										
			9E6D 5 TST 3 SP1										
							9EAE 5 LDHX 2 IX	9EBE 6 LDHX 4 IX2	9ECE 5 LDHX 3 IX1	9EDE 5 LDX 4 SP2	9EEE 4 LDX 3 SP1	9EFE 5 LDHX 3 SP1
			9E6F 6 CLR 3 SP1								9EDF 5 STX 4 SP2	9EEF 4 STX 3 SP1	9EFF 5 STHX 3 SP1

INH	Inherent	REL	Relative	SP1	Stack Pointer, 8-Bit Offset
IMM	Immediate	IX	Indexed, No Offset	SP2	Stack Pointer, 16-Bit Offset
DIR	Direct	IX1	Indexed, 8-Bit Offset	IX+	Indexed, No Offset with Post Increment
EXT	Extended	IX2	Indexed, 16-Bit Offset		
DD	DIR to DIR	IMD	IMM to DIR	IX1+	Indexed, 1-Byte Offset with Post Increment
IX+D	IX+ to DIR	DIX+	DIR to IX+		

Note: All Sheet 2 Opcodes are Preceded by the Page 2 Prebyte (9E)

Prebyte (9E) and Opcode in Hexadecimal

9E60	6	HCS08 Cycles
NEG		Instruction Mnemonic
3	SP1	Addressing Mode

Number of Bytes

Addressing Modes

INHERENT	CLRA	
IMMEDIATE	LDA	#$12
DIRECT	LDA	$50
EXTENDED	LDA	$4000
INDEXED	LDA	30000,X
	LDA	$8,X+
	……	
RELATIVE	BNE	LOOP

INSTRUCTION SET (By Functions)

Data Movement

Arithmetic

Logical

Data Manipulation (shift)

Bit Manipulation

Program Control

Binary Coded Decimal

Special

Data Movement
- Load CPU Registers -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		

LDA #opr
LDA opr
LDA opr
LDA opr,X
LDA opr,X
LDA ,X
LDA opr,SP
LDA opr,SP
	Load Accumulator from Memory	A ← (M)	0	–	–	y	y	–	IMM
									DIR
									EXT
									IX2
									IX1
									IX
									SP1
									SP2
										2
										3
										4
										4
										3
										2
										4
										5

LDX #opr
LDX opr
LDX opr
LDX opr,X
LDX opr,X
LDX ,X
LDX opr,SP
LDX opr,SP
	Load Index Register X from Memory	X ← (M)	0	–	–	y	y	–	IMM
									DIR
									EXT
									IX2
									IX1
									IX
									SP1
									SP2
										2
										3
										4
										4
										3
										2
										4
										5

LDHX #opr LDHX opr	Load Index Register H:X from Memory	H:X ← (M:M + 1)	0	–	–	y	y	–	IMM DIR	3 4

Data Movement
- Store CPU Registers -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
STA opr STA opr STA opr,X STA opr,X STA ,X STA opr,SP STA opr,SP	Store Accumulator in Memory	M ← (A)	0	—	—	—	—	y	DIR EXT IX2 IX1 IX SP1 SP2	3 4 4 3 2 4 5
STX opr STX opr STX opr,X STX opr,X STX ,X STX opr,SP STX opr,SP	Store Index Register X in Memory	M ← (X)	0	—	—	—	—	y	DIR EXT IX2 IX1 IX SP1 SP2	3 4 4 3 2 4 5
STHX opr	Store Index Register H:X in Memory	M:M+1 ← (H:X)	0	—	—	—	—	y	DIR	4

Data Movement
- Stack Operations -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
PSHA	Push Accumulator onto Stack	Push (A); SP <— (SP −$01)	–	–	–	–	–	y	INH	2
PSHH	Push Index Reg H onto Stack	Push (H); SP <— (SP −$01)	–	–	–	–	–	y	INH	2
PSHX	Push Index Reg X onto Stack	Push (X); SP <— (SP −$01)	–	–	–	–	–	y	INH	2
PULA	Pull Accumulator from Stack	SP <— (SP + $01); Pull (A)	–	–	–	–	–	y	INH	2
PULH	Pull Index Reg H from Stack	SP <— (SP + $01); Pull (H)	–	–	–	–	–	y	INH	2
PULX	Pull Index Reg X from Stack	SP <— (SP + $01); Pull (X)	–	–	–	–	–	y	INH	2

Data Movement
- Register to Register -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
TAP	Transfer Accumulator to CCR	CCR $\leftarrow$ (A)	-	-	-	-	-	y	INH	2
TPA	Transfer CCR to Accumulator	A $\leftarrow$ (CCR)	-	-	-	-	-	y	INH	1
TAX	Transfer A to Index X	X $\leftarrow$ (A)	-	-	-	-	-	y	INH	1
TXA	Transfer Index X to A	A $\leftarrow$ (X)	-	-	-	-	-	y	INH	1
TXS	Transfer Index Register to SP	SPH:SP $\leftarrow$ (H:X) -$0001	-	-	-	-	-	y	INH	2
TSX	Transfer SP to Index Register	H:X $\leftarrow$ (SPH:SP) + $0001	-	-	-	-	-	y	INH	2

Data Movement
- Memory to Memory -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		

MOV opr,opr
MOV opr,X+
MOV #opr,opr
MOV X+,opr
	Move	(M)Destination ← (M)Source
		
		H:X ← (H:X) + 1 in X+ modes
			0	-	-	-	-	y	DD
									DIX+
									IMD
									IX+D
										5
										4
										4
										4

See Addressing Modes, Memory to Memory, for MOV instruction usage notes.

Arithmetic
- Addition -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		

ADD #opr
ADD opr
ADD opr
ADD opr,X
ADD opr,X
ADD ,X
ADD opr,SP
ADD opr,SP
	Add Memory to Accumulator without Carry	A <─ (A) + (M)	-	-	-	-	-	y	IMM
									DIR
									EXT
									IX2
									IX1
									IX
									SP1
									SP2
										2
										3
										4
										4
										3
										2
										4
										5

ADC #opr ADC opr ADC opr ADC opr,X ADC opr,X ADC ,X ADC opr,SP ADC opr,SP	Add Memory to Accumulator with Carry	A <─ (A) + (M) + (C)	-	-	-	-	-	y	IMM DIR EXT IX2 IX1 IX SP1 SP2	2 3 4 4 3 2 4 5

Arithmetic
- Subtraction -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		

SUB #opr
SUB opr
SUB opr
SUB opr,X
SUB opr,X
SUB ,X
SUB opr,SP
SUB opr,SP
	Subtract Memory from Accumulator without Carry	$A \leftarrow (A) - (M)$	-	-	-	-	-	y	IMM
									DIR
									EXT
									IX2
									IX1
									IX
									SP1
									SP2
										2
										3
										4
										4
										3
										2
										4
										5

SBC #opr SBC opr SBC opr SBC opr,X SBC opr,X SBC ,X SBC opr,SP SBC opr,SP	Subtract Memory from Accumulator with Carry	$A \leftarrow (A) - (M) - (C)$	-	-	-	-	-	y	IMM DIR EXT IX2 IX1 IX SP1 SP2	2 3 4 4 3 2 4 5

Arithmetic
- Multiplication & Division -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
MUL	Unsigned 8-bit x 8-bit Multiply	X:A <— (X) x (A)	-	0	-	-	-	0	INH	5
DIV	Unsigned 16-bit x 8-bit Divide	A <— (H:A) -(X) H <— Remainder	-	-	-	-	-	y	INH	7

MUL

- **X will contain the MSB of product**
- **A will contain the LSB of product**

DIV

- **H is MSB of dividend**
- **A is LSB of dividend**
- **X is unaffected**

Arithmetic
- Increment & Decrement -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
INC opr INCA INCX INC opr,X INC ,X INC opr,SP	Increment	M <— (M) + 1 A <— (A) + 1 X <— (X) + 1 M <— (M) + 1 M <— (M) + 1 M <— (M) + 1	-	-	-	-	-	y	DIR INH INH IX1 IX SP1	4 1 1 4 3 5
DEC opr DECA DECX DEC opr,X DEC ,X DEC opr,SP	Decrement	M <— (M) -1 A <— (A) -1 X <— (X) -1 M <— (M) -1 M <— (M) -1 M <— (M) -1	-	-	-	-	-	y	DIR INH INH IX1 IX SP1	4 1 1 4 3 5

Arithmetic
- Complement & Negation -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
COM opr COMA COMX COM opr,X COM ,X COM opr,SP	Complement (One's Complement)	M <— $FF -(M) A <— $FF -(A) X <— $FF -(X) M <— $FF -(M) M <— $FF -(M) M <— $FF -(M)	0	-	-	-	-	1	DIR INH INH IX1 IX SP1	4 1 1 4 3 5
NEG opr NEGA NEGX NEG opr,X NEG ,X NEG opr,SP	Negate (Two's Complement)	M <— $00 -(M) A <— $00 -(A) X <— $00 -(X) M <— $00 -(M) M <— $00 -(M) M <— $00 -(M)	-	-	-	-	-	y	DIR INH INH IX1 IX SP1	4 1 1 4 3 5

One's Complement
- **Unsigned operation**

Two's Complement
- **Signed operation**

Arithmetic
- Comparison -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		

CMP #opr
CMP opr
CMP opr
CMP opr,X
CMP opr,X
CMP ,X
CMP opr,SP
CMP opr,SP
	Compare Accumulator with Memory	A - (M)	-	-	-	-	-	y	IMM
									DIR
									EXT
									IX2
									IX1
									IX
									SP1
									SP2
										2
										3
										4
										4
										3
										2
										4
										5

CPX #opr CPX opr CPX opr CPX opr,X CPX opr,X CPX ,X CPX opr,SP CPX opr,SP	Compare Index Register X with Memory	X - (M)	-	-	-	-	-	y	IMM DIR EXT IX2 IX1 IX SP1 SP2	2 3 4 4 3 2 4 5
CPHX #opr CPHX opr	Compare Index Register H:X with Memory	H:X - (M:M + 1)	-	-	-	-	-	y	IMM DIR	3 4

Arithmetic
- Miscellaneous -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
CLR opr CLRA CLRX CLRH CLR opr,X CLR ,X CLR opr,SP	Clear	M <— $00 A <— $00 X <— $00 H <— $00 M <— $00 M <— $00 M <— $00	0	–	–	0	1	–	DIR INH INH INH IX1 IX SP1	3 1 1 1 3 2 4
TST opr TSTA TSTX TST opr,X TST ,X TST opr,SP	Test for Negative or Zero	(M) – $00 (A) – $00 (X) – $00 (M) – $00 (M) – $00 (M) – $00	0	–	–	y	y	–	DIR INH INH IX1 IX SP1	3 1 1 3 2 4
AIS #opr	Add Immediate Value (Signed) to Stack Pointer	SPH:SP <— (SPH:SP) + (16 << M)	–	–	–	–	–	–	IMM	2
AIX #opr	Add Immediate Value (Signed) to Index Register H:X	H:X <— (H:X) + (16 << M)	–	–	–	–	–	–	IMM	2

AIS （Add Imm to Sp）

AIS can be used to quickly allocate or deallocate stack space

- **Temporary variables**
- **Procedure frames**

00F7	X X		← SP
	temp 1		
	temp 2 msb	AIS #-6	
	temp 2 lsb		
	temp 3		
	temp 4	AIS #6	
	temp 5		← SP
	PC msb		
00FF	PC lsb		
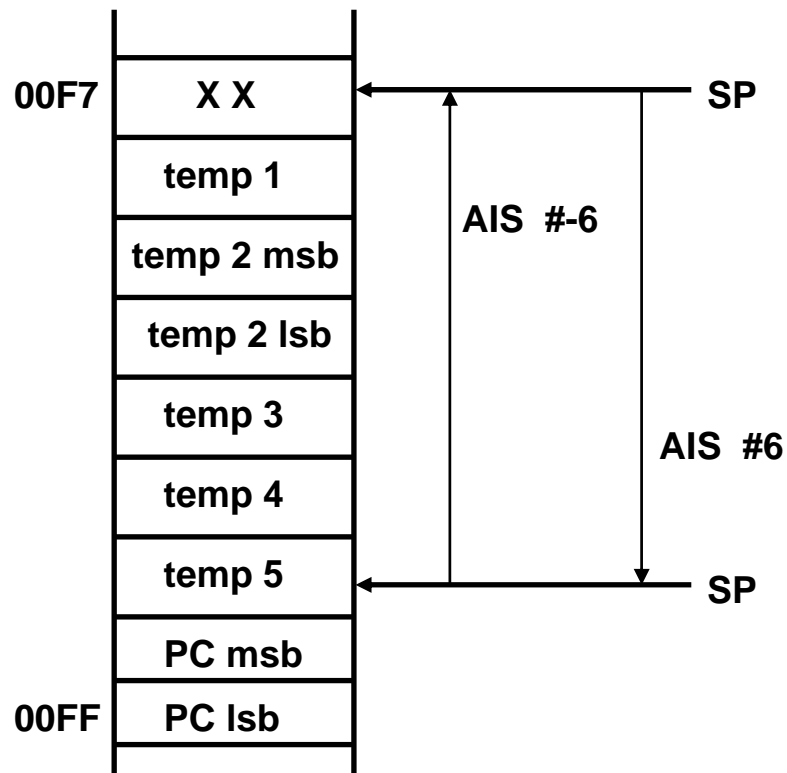
```
SUB1    AIS    #-6      ;Allocate 6 bytes
               •
               •
        AIS    #6       ;Deallocate 6 bytes
        RTS
```

AIX (Add Imm to H:X)

- **Efficiently increment or decrement the H:X register**

  - **AIX   only affect the X register**

  - **INCX / DECX affect the CCR, AIX does not!**

- **INCX / DECX Looping over blocks of memory**

  - **Indexed addressing with post increment only**

- **AIX, Only available for MOV and CBEQ instructions!**

Logical Operators
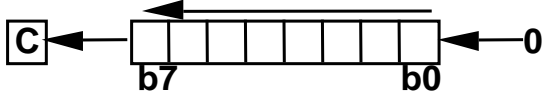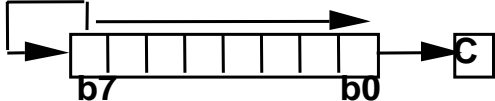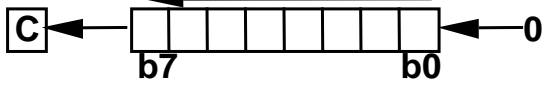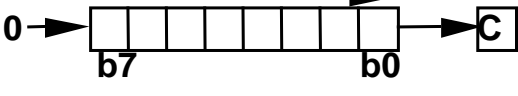
Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
AND #opr AND opr AND opr AND opr,X AND opr,X AND ,X AND opr,SP AND opr,SP	Logical AND Accumulator and Memroy	A <− (A) ∧ (M)	0	-	-	-	-	y	IMM DIR EXT IX2 IX1 IX SP1 SP2	2 3 4 4 3 2 4 5
ORA #opr ORA opr ORA opr ORA opr,X ORA opr,X ORA ,X ORA opr,SP ORA opr,SP	Inclusive OR Accumulator and Memory	A <− (A) + (M)	0	-	-	-	-	y	IMM DIR EXT IX2 IX1 IX SP1 SP2	2 3 4 4 3 2 4 5
EOR #opr EOR opr EOR opr EOR opr,X EOR opr,X EOR ,X EOR opr,SP EOR opr,SP	Exclusive OR Accumulator and Memory	A <− (A) 辨(M)	0	-	-	-	-	y	IMM DIR EXT IX2 IX1 IX SP1 SP2	2 3 4 4 3 2 4 5

Bit Manipulation

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
BIT #opr BIT opr BIT opr BIT opr,X BIT opr,X BIT ,X BIT opr,SP BIT opr,SP	Bit test (AND) Accumulator with Memory	$A \wedge (M)$	0	–	–	y	y	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	2 3 4 4 3 2 4 5
BCLR n,opr	Clear bit n in Memory	$Mn \leftarrow 0$	–	–	–	–	–	–	DIR	4
BSET n,opr	Set bit n in Memory	$Mn \leftarrow 1$	–	–	–	–	–	–	DIR	4
CLC	Clear Carry Bit	$C \leftarrow 0$	–	–	–	–	–	0	INH	1
SEC	Set Carry Bit	$C \leftarrow 1$	–	–	–	–	–	1	INH	1
CLI	Clear Interrupt Mask	$I \leftarrow 0$	–	–	0	–	–	–	INH	2
SEI	Set Interrupt Mask	$I \leftarrow 1$	–	–	1	–	–	–	INH	2

Data Manipulation
- Shifts -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
ASL opr ASLA ASLX ASL opr,X ASL ,X ASL opr,SP	Arithmetic Shift Left (Same as LSL)	C ← [b7...b0] ← 0	-	-	-	-	-	y	DIR INH INH IX1 IX SP1	4 1 1 4 3 5
ASR opr ASRA ASRX ASR opr,X ASR ,X ASR opr,SP	Arithmetic Shift Right	[b7...b0] → C	-	-	-	-	-	y	DIR INH INH IX1 IX SP1	4 1 1 4 3 5
LSL opr LSLA LSLX LSL opr,X LSL ,X LSL opr,SP	Logical Shift Left	C ← [b7...b0] ← 0	-	-	-	-	-	y	DIR INH INH IX1 IX SP1	4 1 1 4 3 5
LSR opr LSRA LSRX LSR opr,X LSR ,X LSR opr,SP	Logical Shift Right	0 → [b7...b0] → C	-	-	-	0	-	y	DIR INH INH IX1 IX SP1	4 1 1 4 3 5

Data Manipulation
- Rotates -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
ROL opr ROLA ROLX ROL opr,X ROL ,X ROL opr,SP	Rotate Left through Carry		-	-	-	-	-	Y	DIR INH INH IX1 IX SP1	4 1 1 4 3 5
ROR opr RORA RORX ROR opr,X ROR ,X ROR opr,SP	Rotate Right through Carry		-	ℂ	-	-	-	Y	DIR INH INH IX1 IX SP1	4 1 1 4 3 5

Program Control
- Branches -

Instruction	Mnemonic
Branch if Carry Clear	BCC
Branch if Carry Set	BCS
Branch if Equal	BEQ
Branch if Greater Than or Equal (Signed)	BGE
Branch if Greater Than (Signed)	BGT
Branch if Half-Carry Clear	BHCC
Branch if Half-Carry Set	BHCS
Branch if Higher	BHI
Branch if Higher or Same	BHS (BCC)
Branch if Interrupt Line High	BIH
Branch if Interrupt Line Low	BIL
Branch if Less Than or Equal (Signed)	BLE

Program Control
- Branches -

Branch if Lower	BLO (BCS)
Branch if Lower or Same	BLS
Branch if Less Than (Signed)	BLT
Branch if Interrupt Mask Clear	BMC
Branch if Minus	BMI
Branch if Interrupt Mask Set	BMS
Branch if Not Equal	BNE
Branch if Plus	BPL
Branch Always	BRA
Branch if Bit n in Memory Clear	BRCLR
Branch if Bit n in Memory Set	BRSET
Branch Never	BRN
Branch to Subroutine	BSR

Program Control
- Branches -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
Bcc rel	Branch if condition is true (CC, CS, HCC, HCS, HI, HS, LO, LS, PL, MI, EQ, NE, GE, GT, LE, LT, IH, IL, MC, MS)	PC $\leftarrow$ (PC) + $0002 + rel - cc	-	-	-	-	-	y	REL	3
BRA rel	Branch Always	PC $\leftarrow$ (PC) + $0002 + rel	-	-	-	-	-	y	REL	3
BRN rel	Branch Never	PC $\leftarrow$ (PC) + $0002	-	-	-	-	-	y	REL	3
BRCLRn,opr,rel	Branch if Bit n in Memory is Clear	PC $\leftarrow$ (PC) + $0003 + rel $\leftarrow$ Mn = 0	-	-	-	-	-	y	DIR/ REL	5
BRSETn,opr,rel	Branch if Bit n in Memory is Set	PC $\leftarrow$ (PC) + $0003 + rel $\leftarrow$ Mn = 1	-	-	-	-	-	y	DIR/ REL	5

Program Control
- Special Branching -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
CBEQ opr,rel		PC ← (PC) + $0003 +rel ? (A) – (M) = $00							DIR	5
CBEQA #opr,rel		PC ← (PC) + $0003 +rel ? (A) – (M) = $00							IMM	4
CBEQX #opr,rel		PC ← (PC) + $0003 +rel ? (X) – (M) = $00							IMM	4
CBEQ X+,rel	Compare and Branch if Equal	PC ← (PC) + $0003 +rel ? (A) – (M) = $00	–	–	–	–	–	–	IX+	4
CBEQ opr,X+,rel		PC ← (PC) + $0002 +rel ? (A) – (M) = $00							IX1+	5
CBEQ opr,SP,rel		PC ← (PC) + $0004 +rel ? (A) – (M) = $00							SP1	6
DBNZ opr,rel		M ← (M) – $01 PC ← (PC) + $0003 + rel ? (M) ° 0							DIR	5
DBNZA rel		A ← (A) – $01 PC ← (PC) + $0002 + rel ? (A) ° 0							INH	3
DBNZX rel	Decrement and Branch if not Zero	X ← (X) – $01 PC ← (PC) + $0002 + rel ? (X) ° 0	–	–	–	–	–	–	INH	3
DBNZ X,rel		M ← (M) – $01 PC ← (PC) + $0002 + rel ? (M) ° 0							IX	4
DBNZ opr,X,rel		M ← (M) – $01 PC ← (PC) + $0003 + rel ? (M) ° 0							IX1	5
DBNZ opr,SP,rel		M ← (M) – $01 PC ← (PC) + $0004 + rel ? (M) ° 0							SP1	6

CBEQ and DBNZ

CBEQ combines the CMP and BEQ instruction

- **Faster table lookup/search operations**

DBNZ combines the DEC and BNE instructions

- **Faster more efficient looping**

CBEQ Example

* Subroutine that searches a string for the next blank character
* and then points the H:X register to the character immediately
* following the blank.
* H:X is assumed to already point to a location in the string.

```
            ORG     $00A0
String      RMB     50              ;The character string


            ORG     $6E00
 Search     LDA     #$20            ;Load search character

 Loop       CBEQ    X+,Out          ;Match?
            BRA     Loop            ;No match, do it again.
```

* X post increment will occur regardless of whether the branch is
* taken. Therefore when a match is found H:X already points to the
* next character.

```
 Out        RTS
```

How could you change this example to avoid searching past the
end of the string?

DBNZ Example

```
* Time delay routine
* Delay = N x (160.0+0.375) µs for an 8 MHz CPU clock
* For example, for delay = 10ms  N = 63


N              EQU      63                 ;Loop counter for 10 ms delay

               ORG      $50
Count          RMB      1                  ;Loop counter

               ORG      $6E00
Delay          LDA      #N                 ;Set delay constant
Loop           DBNZ     Count,Loop         ;Inner loop, Count starts at $00
               DBNZA    Loop
               RTS
```

How were the values 160.0 and 0.375 derived?

Program Control
- Jumps and Subroutines -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
JMP opr JMP opr JMP opr,X JMP opr,X JMP ,X	Jump to location	PC <— Jump Address	-	-	-	-	-	y	DIR EXT IX2 IX1 IX	2 3 4 3 2
JSR opr JSR opr JSR opr,X JSR opr,X JSR ,X	Jump to subroutine	PC <— (PC) + n (n=1, 2 or 3) Push (PCL); SP <— (SP) -1 Push (PCH); SP <— (SP) -1 PC <— Unconditional Address	-	-	-	-	-	y	DIR EXT IX2 IX1 IX	4 5 6 5 4
BSR rel	Branch to subroutine	PC <— (PC) + 2 Push (PCL); SP <— (SP) -1 Push (PCH); SP <— (SP) -1 PC <— (PC) + rel	-	-	-	-	-	y	REL	4
RTS	Return from subroutine	SP <— (SP) + 1; Pull (PCH) SP <— (SP) + 1; Pull (PCL)	-	-	-	-	-	y	INH	4

Program Control
- Interrupt -

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
SWI	Software Interrupt	PC <− (PC) + 1 Push (PCL); SP <− (SP) -1 Push (PCH); SP <− (SP) -1 Push (X); SP <− (SP) -1 Push (A); SP <− (SP) -1 Push (CCR); SP <− (SP) -1 I <− 1 PCH <− Interrupt Vector High Byte PCL <− Interrupt Vector Low Byte	-	-	1	-	-	y	INH	9
RTI	Return from Interrupt	SP <− (SP) + 1; Pull (CCR) SP <− (SP) + 1; Pull (A) SP <− (SP) + 1; Pull (X) SP <− (SP) + 1; Pull (PCH) SP <− (SP) + 1; Pull (PCL)	-	-	-	-	-	y	INH	7

Binary Coded Decimal

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
DAA	Decimal Adjust Accumulator	$(A)_{10}$	U	–	–	y	y	y	INH	2
NSA	Nibble Swap contents of Accumulator	A ⟵ (A[3:0]:A[7:4])	–	–	–	–	–	–	INH	3

Special Instructions

Source Forms	Description	Operation	Effect on CCR						Address Modes	Bus Cycles
			V	H	I	N	Z	C		
RSP	Reset Stack Pointer	SPL ⟵ $FF	–	–	–	–	–	–	INH	1
NOP	No Operation	None	–	–	–	–	–	–	INH	1
STOP	Stop Processor and wait for interrupt	I ⟵ 0 Stop Oscillator	–	–	0	–	–	–	INH	1
WAIT	Halt Processor and wait for interrupt	I ⟵ 0	–	–	0	–	–	–	INH	1

WAIT:

- **CPU08 stops processing instructions**
- **waits for an interrupt**

STOP:

- **CPU08 stops processing instructions**
- **Stops the oscillator circuit**
  - **Puts MPU in low power state**
- **Waits for interrupt**