Document No : UniKL MFI_SD_AC41
Revision No: 02
Effective Date: 01 December 2008

CONFIDENTIAL

SET A

UNIVERSITI KUALA LUMPUR
Malaysia France Institute

FINAL EXAMINATION
JANUARY 2011 SESSION

SUBJECT CODE	:	FSD 23102
SUBJECT TITLE	:	MICROPROCESSOR
LEVEL	:	DIPLOMA
TIME / DURATION	:	12.30pm – 2.30pm (2 HOURS)
DATE	:	04 MAY 2011

INSTRUCTIONS TO CANDIDATES

1. Please read the instructions given in the question paper CAREFULLY.
2. This question paper is printed on both sides of the paper.
3. Please write your answers on the answer booklet provided.
4. Answer should be written in blue or black ink except for sketching, graphic and illustration.
5. This question paper consists of TWO (2) sections. Section A and B. Answer all questions in Section A. For Section B, answer two (2) questions only.
6. Answer all questions in English.
7. All types of calculators are STRICTLY PROHIBITED during the examination.

THERE ARE 6 PAGES OF QUESTIONS AND 4 PAGES OF APPENDICES, EXCLUDING THIS PAGE.

SECTION A (Total: 60 marks)

INSTRUCTION: Answer ALL questions.
Please use the answer booklet provided.

Question 1

(a) Name two (2) microprocessor based applications system and give two (2) examples for each of them.

(4 marks)

(b) List three (3) types of microprocessor programming language.

(3 marks)

(c) Write the meaning of program in microprocessor system.

(2 marks)

(d) Assume a data, $17 is stored in memory location $0446. Find the final data value that contained in memory address $5000.

```
LDAA    $0446
LDAB    #$15
SBA
STAA    $5000
END
```

(2 marks)

(e) ____________________ is used to illustrate in *diagrammatical form* of the steps in the program.

(1 mark)

(f) State the sequence of source statements in assembly language format.

(3 marks)

(g) List three (3) microprocessor pins that classified as three-state.

(3 marks)

(h) In BPL instruction, it will causes branch if ______________ and proceed to next instruction in program if ________________.

(2 marks)

Question 2

Convert and perform the arithmetic operation below. You are required to show the conversion procedure algorithmically.

(a) Convert **&679** to hexadecimal form.

(3 marks)

(b) Convert **%11010011** to decimal form.

(3 marks)

(c) Convert two unsigned hexadecimal numbers $AB and $13 into binary form and perform the binary number addition for these two numbers ($AB+$13).

(4 marks)

(d) By using two's complement, perform the following operation.

Note: *Your calculations should be in 8-bit integer numbers.*

&70 - $1F

(6 marks)

(e) By referring to Question 2(d), what happens to the status of C-bit and H-bit in CCR register and justify your answer.

(4 marks)

Question 3

(a) Explain the function of each register below:

(i) Memory Address Register (MAR)

(2 marks)

(ii) Stack Pointer (SP)

(2 marks)

(b) Identify the lowest priority of Interrupt Signal in Microprocessor 6809 and describe briefly on this.

(4 marks)

(c) What is the purpose of Accumulator?

(2 marks)

(d) Explain briefly the clock signals use in MC6809 microprocessor. Draw the timing diagram of the clock signals.

(5 marks)

(e) List down the principle operation for MC6809 under control of Φ1 and Φ2 clocks.

(1 mark)

(e) Explain briefly about Address/Data bus in MC6809.

(4 marks)

SECTION B (Total: 40 marks)

INSTRUCTION: Answer TWO (2) questions only

Please use the answer booklet provided.

Question 4

(a) Define the meaning of these three arithmetic instructions as stated below,

(i) SUBA

(ii) SBA

(iii) SBCA

(6 marks)

(b) Design a flow chart to subtract $53AB from the data of memory location $0500 and $0501. The data are two bytes and locations $0501 forms the LSB. The result is saved in two locations starting at $0300 and $0301. Location $0300 saves the MSB result.

(4 marks)

(c) Write an assembly language program, complete with comments to represent your algorithm written in Question 4 (b). The program code also must be started at location $0200.

Note: *Refer to Instruction Set in Appendices.*

(10 marks)

Question 5

(a) Explain briefly Direct Addressing Mode.

(4 marks)

(b) Write an assembly language and the equivalent machine code to add data in Accumulator A with data value that is stored in memory location address $003E by using Direct Addressing Mode.

Note: *Refer to Instruction Set in Appendices.*

(4 marks)

(c) By referring Question 5 (b) above, complete the diagram below in **Figure 1** (highlighted in grey box) after execution of the instruction. Assume that the op-code and the operand of program are kept at memory location $4001 to $4002 and Accumulator A stored the value $56 before execution.

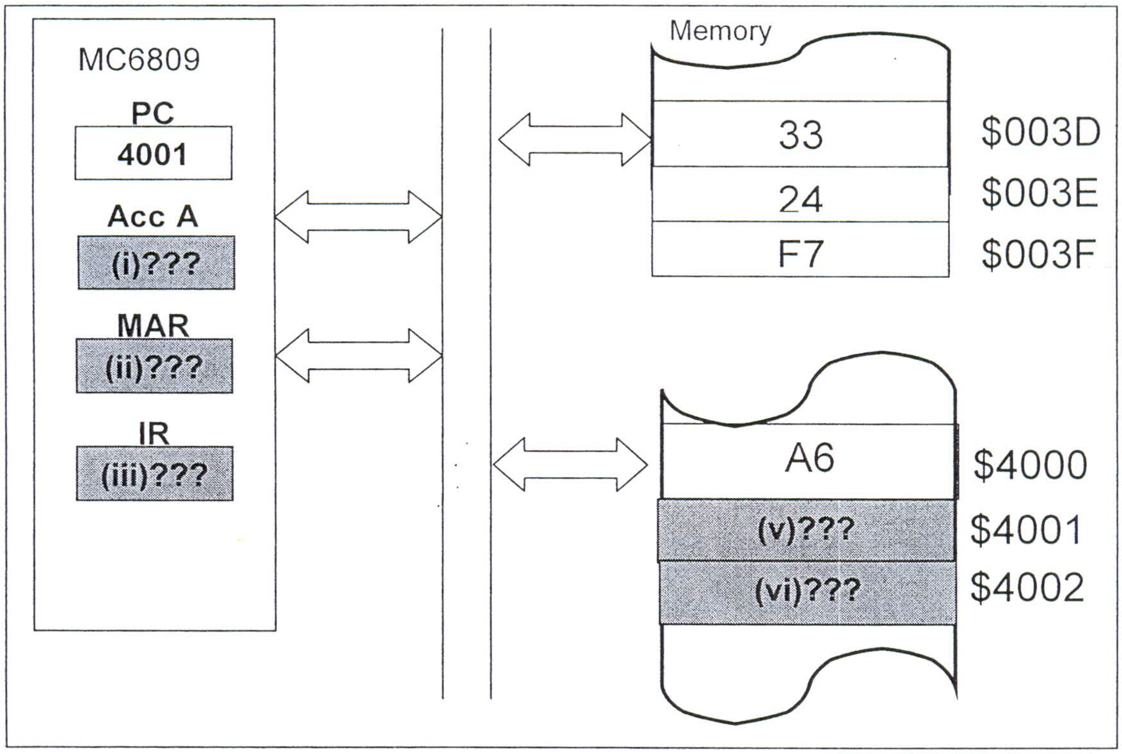


Figure 1 : Direct Addressing Mode after execution process.

(6 marks)

(d) How many bytes are required in performing the task?

(2 marks)

(e) Compute the machine cycles needed for the instruction above and calculate the time in second. Assume the crystal clock is 2 MHz.

(4 marks)

Question 6

(a) Describe briefly on the Relative Addressing Modes.

(3 marks)

(b) Design a flow chart that squares the numbers from 1 to 9 and the each result is saved in locations $0301 to $0309. The result for square of 1 is saved in location $0301 and the last memory location, $0309 holds the result for squaring of 9.

(5 marks)

(c) Write an assembly language program, complete with comments to represent your algorithm written in Question 6 (a). The program also must be started at location $0200.

Note: *Refer to Instruction Set in Appendices.*

(12 marks)

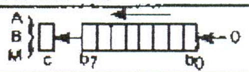
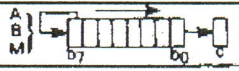
END OF QUESTION

APPENDICES

APPENDIX - INSTRUCTION SET

MC6809

PROGRAMMING AID

Instruction	Forms	Addressing Modes															Description	5	3	2	1	0
		Immediate			Direct			Indexed			Extended			Inherent								
		Op	~	#	Op	~	#	Op	~	#	Op	~	#	Op	~	#		H	N	Z	V	C
ABX														3A	3	1	B + X → X (Unsigned)	•	•	•	•	•
ADC	ADCA	89	2	2	99	4	2	A9	4+	2+	B9	5	3				A + M + C → A	↕	↕	↕	↕	↕
	ADCB	C9	2	2	D9	4	2	E9	4+	2+	F9	5	3				B + M + C → B	↕	↕	↕	↕	↕
ADD	ADDA	8B	2	2	9B	4	2	AB	4+	2+	BB	5	3				A + M → A	↕	↕	↕	↕	↕
	ADDB	CB	2	2	DB	4	2	EB	4+	2+	FB	5	3				B + M → B	↕	↕	↕	↕	↕
	ADDD	C3	4	3	D3	6	2	E3	6+	2+	F3	7	3				D + M:M + 1 → D	•	↕	↕	↕	↕
AND	ANDA	84	2	2	94	4	2	A4	4+	2+	B4	5	3				A ∧ M → A	•	↕	↕	0	•
	ANDB	C4	2	2	D4	4	2	E4	4+	2+	F4	5	3				B ∧ M → B	•	↕	↕	0	•
	ANDCC	1C	3	2													CC ∧ IMM → CC					7
ASL	ASLA													48	2	1		8	↕	↕	↕	↕
	ASLB													58	2	1		8	↕	↕	↕	↕
	ASL				08	6	2	68	6+	2+	78	7	3					8	↕	↕	↕	↕
ASR	ASRA													47	2	1		8	↕	↕	•	↕
	ASRB													57	2	1		8	↕	↕	•	↕
	ASR				07	6	2	67	6+	2+	77	7	3					8	↕	↕	•	↕
BIT	BITA	85	2	2	95	4	2	A5	4+	2+	B5	5	3				Bit Test A (M ∧ A)	•	↕	↕	0	•
	BITB	C5	2	2	D5	4	2	E5	4+	2+	F5	5	3				Bit Test B (M ∧ B)	•	↕	↕	0	•
CLR	CLRA													4F	2	1	0 → A	•	0	1	0	0
	CLRB													5F	2	1	0 → B	•	0	1	0	0
	CLR				0F	6	2	6F	6+	2+	7F	7	3				0 → M	•	0	1	0	0
CMP	CMPA	81	2	2	91	4	2	A1	4+	2+	B1	5	3				Compare M from A	8	↕	↕	↕	↕
	CMPB	C1	2	2	D1	4	2	E1	4+	2+	F1	5	3				Compare M from B	8	↕	↕	↕	↕
	CMPD	10 83	5	4	10 93	7	3	10 A3	7+	3+	10 B3	8	4				Compare M:M + 1 from D	•	↕	↕	↕	↕
	CMPS	11 8C	5	4	11 9C	7	3	11 AC	7+	3+	11 BC	8	4				Compare M:M + 1 from S	•	↕	↕	↕	↕
	CMPU	11 83	5	4	11 93	7	3	11 A3	7+	3+	11 B3	8	4				Compare M:M + 1 from U	•	↕	↕	↕	↕
	CMPX	8C	4	3	9C	6	2	AC	6+	2+	BC	7	3				Compare M:M + 1 from X	•	↕	↕	↕	↕
	CMPY	10 8C	5	4	10 9C	7	3	10 AC	7+	3+	10 BC	8	4				Compare M:M + 1 from Y	•	↕	↕	↕	↕
COM	COMA													43	2	1	$\overline{A}$ → A	•	↕	↕	0	1
	COMB													53	2	1	$\overline{B}$ → B	•	↕	↕	0	1
	COM				03	6	2	63	6+	2+	73	7	3				$\overline{M}$ → M	•	↕	↕	0	1
CWAI		3C	≥20	2													CC ∧ IMM → CC Wait for Interrupt					7
DAA														19	2	1	Decimal Adjust A	•	↕	↕	0	↕
DEC	DECA													4A	2	1	A − 1 → A	•	↕	↕	↕	•
	DECB													5A	2	1	B − 1 → B	•	↕	↕	↕	•
	DEC				0A	6	2	6A	6+	2+	7A	7	3				M − 1 → M	•	↕	↕	↕	•
EOR	EORA	88	2	2	98	4	2	A8	4+	2+	B8	5	3				A ⊻ M → A	•	↕	↕	0	•
	EORB	C8	2	2	D8	4	2	E8	4+	2+	F8	5	3				B ⊻ M → B	•	↕	↕	0	•
EXG	R1, R2	1E	8	2													R1 ↔ R2[2]	•	•	•	•	•
INC	INCA													4C	2	1	A + 1 → A	•	↕	↕	↕	•
	INCB													5C	2	1	B + 1 → B	•	↕	↕	↕	•
	INC				0C	6	2	6C	6+	2+	7C	7	3				M + 1 → M	•	↕	↕	↕	•
JMP					0E	3	2	6E	3+	2+	7E	4	3				EA[3] → PC	•	•	•	•	•
JSR					9D	7	2	AD	7+	2+	BD	8	3				Jump to Subroutine	•	•	•	•	•
LD	LDA	86	2	2	96	4	2	A6	4+	2+	B6	5	3				M → A	•	↕	↕	0	•
	LDB	C6	2	2	D6	4	2	E6	4+	2+	F6	5	3				M → B	•	↕	↕	0	•
	LDD	CC	3	3	DC	5	2	EC	5+	2+	FC	6	3				M:M + 1 → D	•	↕	↕	0	•
	LDS	10 CE	4	4	10 DE	6	3	10 EE	6+	3+	10 FE	7	4				M:M + 1 → S	•	↕	↕	0	•
	LDU	CE	3	3	DE	5	2	EE	5+	2+	FE	6	3				M:M + 1 → U	•	↕	↕	0	•
	LDX	8E	3	3	9E	5	2	AE	5+	2+	BE	6	3				M:M + 1 → X	•	↕	↕	0	•
	LDY	10 8E	4	4	10 9E	6	3	10 AE	6+	3+	10 BE	7	4				M:M + 1 → Y	•	↕	↕	0	•
LEA	LEAS							32	4+	2+							EA[3] → S	•	•	•	•	•
	LEAU							33	4+	2+							EA[3] → U	•	•	•	•	•
	LEAX							30	4+	2+							EA[3] → X	•	•	↕	•	•
	LEAY							31	4+	2+							EA[3] → Y	•	•	↕	•	•

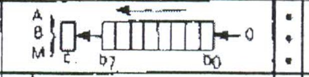
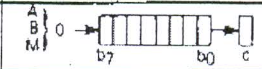
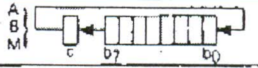
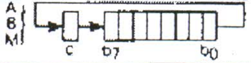
LEGEND:

OP	Operation Code (Hexadecimal)
~	Number of MPU Cycles
#	Number of Program Bytes
+	Arithmetic Plus
−	Arithmetic Minus
•	Multiply
$\overline{M}$	Complement of M
→	Transfer Into
H	Half-carry (from bit 3)
N	Negative (sign bit)
Z	Zero result
V	Overflow, 2's complement
C	Carry from ALU
↕	Test and set if true, cleared otherwise
•	Not Affected
CC	Condition Code Register
:	Concatenation
∨	Logical or
∧	Logical and
⊻	Logical Exclusive or

APPENDIX - INSTRUCTION SET

MC6809

PROGRAMMING AID (CONTINUED)

		Addressing Modes																				
		Immediate			Direct			Indexed[1]			Extended			Inherent				5	3	2	1	0
Instruction	Forms	Op	~	#	Op	~	#	Op	~	#	Op	~	#	Op	~	#	Description	H	N	Z	V	C
LSL	LSLA													48	2	1		•	↕	↕	↕	↕
	LSLB													58	2	1		•	↕	↕	↕	↕
	LSL				08	6	2	68	6+	2+	78	7	3					•	↕	↕	↕	↕
LSR	LSRA													44	2	1		•	0	↕	•	↕
	LSRB													54	2	1		•	0	↕	•	↕
	LSR				04	6	2	64	6+	2+	74	7	3					•	0	↕	•	↕
MUL														3D	11	1	A × B → D (Unsigned)	•	•	↕	•	9
NEG	NEGA													40	2	1	$\overline{A}$ + 1 → A	8	↕	↕	↕	↕
	NEGB													50	2	1	$\overline{B}$ + 1 → B	8	↕	↕	↕	↕
	NEG				00	6	2	60	6+	2+	70	7	3				$\overline{M}$ + 1 → M	8	↕	↕	↕	↕
NOP														12	2	1	No Operation	•	•	•	•	•
OR	ORA	8A	2	2	9A	4	2	AA	4+	2+	BA	5	3				A ∨ M → A	•	↕	↕	0	•
	ORB	CA	2	2	DA	4	2	EA	4+	2+	FA	5	3				B ∨ M → B	•	↕	↕	0	•
	ORCC	1A	3	2													CC ∨ IMM → CC				7	
PSH	PSHS	34	5+[4]	2													Push Registers on S Stack	•	•	•	•	•
	PSHU	36	5+[4]	2													Push Registers on U Stack	•	•	•	•	•
PUL	PULS	35	5+[4]	2													Pull Registers from S Stack	•	•	•	•	•
	PULU	37	5+[4]	2													Pull Registers from U Stack	•	•	•	•	•
ROL	ROLA													49	2	1		•	↕	↕	↕	↕
	ROLB													59	2	1		•	↕	↕	↕	↕
	ROL				09	6	2	69	6+	2+	79	7	3					•	↕	↕	↕	↕
ROR	RORA													46	2	1		•	↕	↕	•	↕
	RORB													56	2	1		•	↕	↕	•	↕
	ROR				06	6	2	66	6+	2+	76	7	3					•	↕	↕	•	↕
RTI														3B	6, 15	1	Return From Interrupt					7
RTS														39	5	1	Return from Subroutine	•	•	•	•	•
SBC	SBCA	82	2	2	92	4	2	A2	4+	2+	B2	5	3				A − M − C → A	8	↕	↕	↕	↕
	SBCB	C2	2	2	D2	4	2	E2	4+	2+	F2	5	3				B − M − C → B	8	↕	↕	↕	↕
SEX														1D	2	1	Sign Extend B into A	•	↕	↕	0	•
ST	STA				97	4	2	A7	4+	2+	B7	5	3				A → M	•	↕	↕	0	•
	STB				D7	4	2	E7	4+	2+	F7	5	3				B → M	•	↕	↕	0	•
	STD				DD	5	2	ED	5+	2+	FD	6	3				D → M M + 1	•	↕	↕	0	•
	STS				10 DF	6	3	10 EF	6+	3+	10 FF	7	4				S → M M + 1	•	↕	↕	0	•
	STU				DF	5	2	EF	5+	2+	FF	6	3				U → M M + 1	•	↕	↕	0	•
	STX				9F	5	2	AF	5+	2+	BF	6	3				X → M M + 1	•	↕	↕	0	•
	STY				10 9F	6	3	10 AF	6+	3+	10 BF	7	4				Y → M M + 1	•	↕	↕	0	•
SUB	SUBA	80	2	2	90	4	2	A0	4+	2+	B0	5	3				A − M → A	8	↕	↕	↕	↕
	SUBB	C0	2	2	D0	4	2	E0	4+	2+	F0	5	3				B − M → B	8	↕	↕	↕	↕
	SUBD	83	4	3	93	6	2	A3	6+	2+	B3	7	3				D − M M + 1 → D	•	↕	↕	↕	↕
SWI	SWI[6]													3F	19	1	Software Interrupt 1	•	•	•	•	•
	SWI2[6]													10 3F	20	2	Software Interrupt 2	•	•	•	•	•
	SWI3[6]													11 3F	20	1	Software Interrupt 3	•	•	•	•	•
SYNC														13	≥4	1	Synchronize to Interrupt	•	•	•	•	•
TFR	R1, R2	1F	6	2													R1 → R2[2]	•	•	•	•	•
TST	TSTA													4D	2	1	Test A	•	↕	↕	0	•
	TSTB													5D	2	1	Test B	•	↕	↕	0	•
	TST				0D	6	2	6D	6+	2+	7D	7	3				Test M	•	↕	↕	0	•

NOTES:

1. This column gives a base cycle and byte count. To obtain total count, add the values obtained from the INDEXED ADDRESSING MODE table, Table 2.
2. R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers.
   The 8 bit registers are: A, B, CC, DP
   The 16 bit registers are: X, Y, U, S, D, PC
3. EA is the effective address.
4. The PSH and PUL instructions require 5 cycles plus 1 cycle for each byte pushed or pulled.
5. 5(6) means: 5 cycles if branch not taken, 6 cycles if taken (Branch instructions).
6. SWI sets I and F bits. SWI2 and SWI3 do not affect I and F.
7. Conditions Codes set as a direct result of the instruction.
8. Vaue of half-carry flag is undefined.
9. Special Case — Carry set if b7 is SET.

APPENDIX - INSTRUCTION SET

MC6809

PROGRAMMING AID (CONTINUED)

Branch Instructions

Instruction	Forms	Addressing Mode Relative			Description	5	3	2	1	0
		OP	~5	#		H	N	Z	V	C
BCC	BCC LBCC	24 10 24	3 5(6)	2 4	Branch C=0 Long Branch C=0	• •	• •	• •	• •	• •
BCS	BCS LBCS	25 10 25	3 5(6)	2 4	Branch C=1 Long Branch C=1	• •	• •	• •	• •	• •
BEQ	BEQ LBEQ	27 10 27	3 5(6)	2 4	Branch Z=1 Long Branch Z=0	• •	• •	• •	• •	• •
BGE	BGE LBGE	2C 10 2C	3 5(6)	2 4	Branch ≥ Zero Long Branch ≥ Zero	• •	• •	• •	• •	• •
BGT	BGT LBGT	2E 10 2E	3 5(6)	2 4	Branch > Zero Long Branch > Zero	• •	• •	• •	• •	• •
BHI	BHI LBHI	22 10 22	3 5(6)	2 4	Branch Higher Long Branch Higher	• •	• •	• •	• •	• •
BHS	BHS LBHS	24 10 24	3 5(6)	2 4	Branch Higher or Same Long Branch Higher or Same	• •	• •	• •	• •	• •
BLE	BLE LBLE	2F 10 2F	3 5(6)	2 4	Branch ≤ Zero Long Branch ≤ Zero	• •	• •	• •	• •	• •
BLO	BLO LBLO	25 10 25	3 5(6)	2 4	Branch lower Long Branch Lower	• •	• •	• •	• •	• •

Instruction	Forms	Addressing Mode Relative			Description	5	3	2	1	0
		OP	~5	#		H	N	Z	V	C
BLS	BLS LBLS	23 10 23	3 5(6)	2 4	Branch Lower or Same Long Branch Lower or Same	• •	• •	• •	• •	• •
BLT	BLT LBLT	2D 10 2D	3 5(6)	2 4	Branch < Zero Long Branch < Zero	• •	• •	• •	• •	• •
BMI	BMI LBMI	2B 10 2B	3 5(6)	2 4	Branch Minus Long Branch Minus	• •	• •	• •	• •	• •
BNE	BNE LBNE	26 10 26	3 5(6)	2 4	Branch Z=0 Long Branch Z≠0	• •	• •	• •	• •	• •
BPL	BPL LBPL	2A 10 2A	3 5(6)	2 4	Branch Plus Long Branch Plus	• •	• •	• •	• •	• •
BRA	BRA LBRA	20 16	3 5	2 3	Branch Always Long Branch Always	• •	• •	• •	• •	• •
BRN	BRN LBRN	21 10 21	3 5	2 4	Branch Never Long Branch Never	• •	• •	• •	• •	• •
BSR	BSR LBSR	8D 17	7 9	2 3	Branch to Subroutine Long Branch to Subroutine	• •	• •	• •	• •	• •
BVC	BVC LBVC	28 10 28	3 5(6)	2 4	Branch V=0 Long Branch V=0	• •	• •	• •	• •	• •
BVS	BVS LBVS	29 10 29	3 5(6)	2 4	Branch V=1 Long Branch V=1	• •	• •	• •	• •	• •

SIMPLE BRANCHES

	OP	~	#
BRA	20	3	2
LBRA	16	5	3
BRN	21	3	2
LBRN	1021	5	4
BSR	8D	7	2
LBSR	17	9	3

SIMPLE CONDITIONAL BRANCHES (Notes 1-4)

Test	True	OP	False	OP
N=1	BMI	2B	BPL	2A
Z=1	BEQ	27	BNE	26
V=1	BVS	29	BVC	28
C=1	BCS	25	BCC	24

SIGNED CONDITIONAL BRANCHES (Notes 1-4)

Test	True	OP	False	OP
r>m	BGT	2E	BLE	2F
r≥m	BGE	2C	BLT	2D
r=m	BEQ	27	BNE	26
r≤m	BLE	2F	BGT	2E
r<m	BLT	2D	BGE	2C

UNSIGNED CONDITIONAL BRANCHES (Notes 1-4)

Test	True	OP	False	OP
r>m	BHI	22	BLS	23
r≥m	BHS	24	BLO	25
r=m	BEQ	27	BNE	26
r≤m	BLS	23	BHI	22
r<m	BLO	25	BHS	24

NOTES:

1. All conditional branches have both short and long variations.
2. All short branches are two bytes and require three cycles.
3. All conditional long branches are formed by prefixing the short branch opcode with $10 and using a 16-bit destination offset.
4. All conditional long branches require four bytes and six cycles if the branch is taken or five cycles if the branch is not taken.