

## CS 110 Midterm Solutions

---

### Problem 1 – Some Assembly (and Disassembly) Required (35 points)

Assembly	Machine Code (in HEX)
1. MOVE.L A6, -(A7)	0x2F0E
2. LSL.W #1, D0	0xE348
3. LEA -20(A5, D0.W), A3	0x47F5 00EC

Machine Code	Hint	Assembly
4. 0x2C4F	MOVEA	MOVEA.L A7, A6
5. 0x514F	SUBQ	SUBQ.W #8, A7
6. 0xB7F9 B7F9 B7F8	CMPA	CMPA.L 0xB7F9B7F8, A3

Bonus (+1 point):

LINK A6, #-8

### Problem 2 – Condition Codes and Branching (30 points)

Instruction	D0	D1	A0	N	Z	V	C	BLS	BLE
CMP.L D0, D1	-	-	-	1	0	0	0	N	Y
SUB.L D0, D1	-	0xFDCA 8765	-	-	-	-	-	-	-
CMP.B D1, D0	-	-	-	1	0	0	1	Y	Y
MOVEA.L D1, A0	-	-	0xFDCA 8765	-	-	-	-	-	-

### Problem 3 – Tables (30 points)

```
switch (x) {
  case 13:
  case 16:
    y = 2 * x;
    break;
  case 12:
  case 14:
  case 15:
    y = -1;
    break;
  default:
    y = x;
}
```

a) Write the code for the cases:

```
@caseA:    MOVE.W    D0, D1
           LSL.W    #1, D1
           BRA     @done
@caseB:    MOVE.W    #-1, D1
           BRA     @done
@default:  MOVE.W    D0, D1
@done     ...
```

b) Specify the table (use Metrowerks syntax):

```
CaseTable: DC.W      @caseB
           DC.W      @caseA+2
           DC.W      @caseB+4
           DC.W      @caseB+6
           DC.W      @caseA+8
```

c) Figure out where to jump to and jump there:

```
    CMP.W    #12, D0
    BLT     @default
    CMP.W    #16, D0
    BGT     @default

    MOVE.W   D0, D1
    SUB.W    #12, D1
    LSL.W    #1, D1
    MOVE.W   CaseTable(D1.W), D1
    LEA     CaseTable(D1.W), A0

    JMP     (A0)
```

### Problem 4 – Addressing Modes (25 points)

Given the following initial conditions, show the destination address (or register) and 32-bit value (in hex) **after the execution of these instructions**. Note: unlike problem #2, **every instruction is executed independently** with the following initial conditions:

0000 0000 to 0001 B894	????
0001 B896	0725
0001 B898	5303
0001 B89A	0195
0001 B89C	0000
0001 B89E to FFFE FFFE	DEAD
FFFF 0000 to FFFF FA1C	????
FFFF FA1E	00BF
FFFF FA20	0800
FFFF FA22	0000
FFFF FA24	0000
FFFF FA26 to FFFF FFFE	????

D0	9899 0004
D1	2300 FFFE
A1	10C6 F52C
A2	0001 B89A
A3	0001 C6CD
A4	0001 D2A4
A5	0001 B8B0
A6	0001 C6BA
A7	0001 C5EE
PC	0000 7000

```
#define      kPolMask                0x01
#define      LCD_PolReg              0xFFFFFA21
static      Byte gOldVal;           // located -(0x1A) off A5
...
pattern:   DC.W                    0x0000, 0x0000, 0xAAAA, 0xAAAA, 0xFFFF
           DC.W                    0x0000, 0x5555, 0x5555, 0xFFFF, 0xFFFF
```

	destination EA (or register) in hex.	32-bit hex value beginning at EA (after instruction).
MOVE.B LCD_PolReg, gOldVal	0x0001 B896	0x0025 5303
ORI.B #kPolMask, LCD_PolReg	0xFFFF FA21	0x0100 0000
MOVE.L D1, -(A7)	0x0001 C5EA	0x2300 FFFE
MOVE.W pattern(D0.W), D1	register D1	0x2300 AAAA

...**REMEMBER:** even though these instructions do make sense together, you should execute them all independently (all starting with the initial conditions given above).