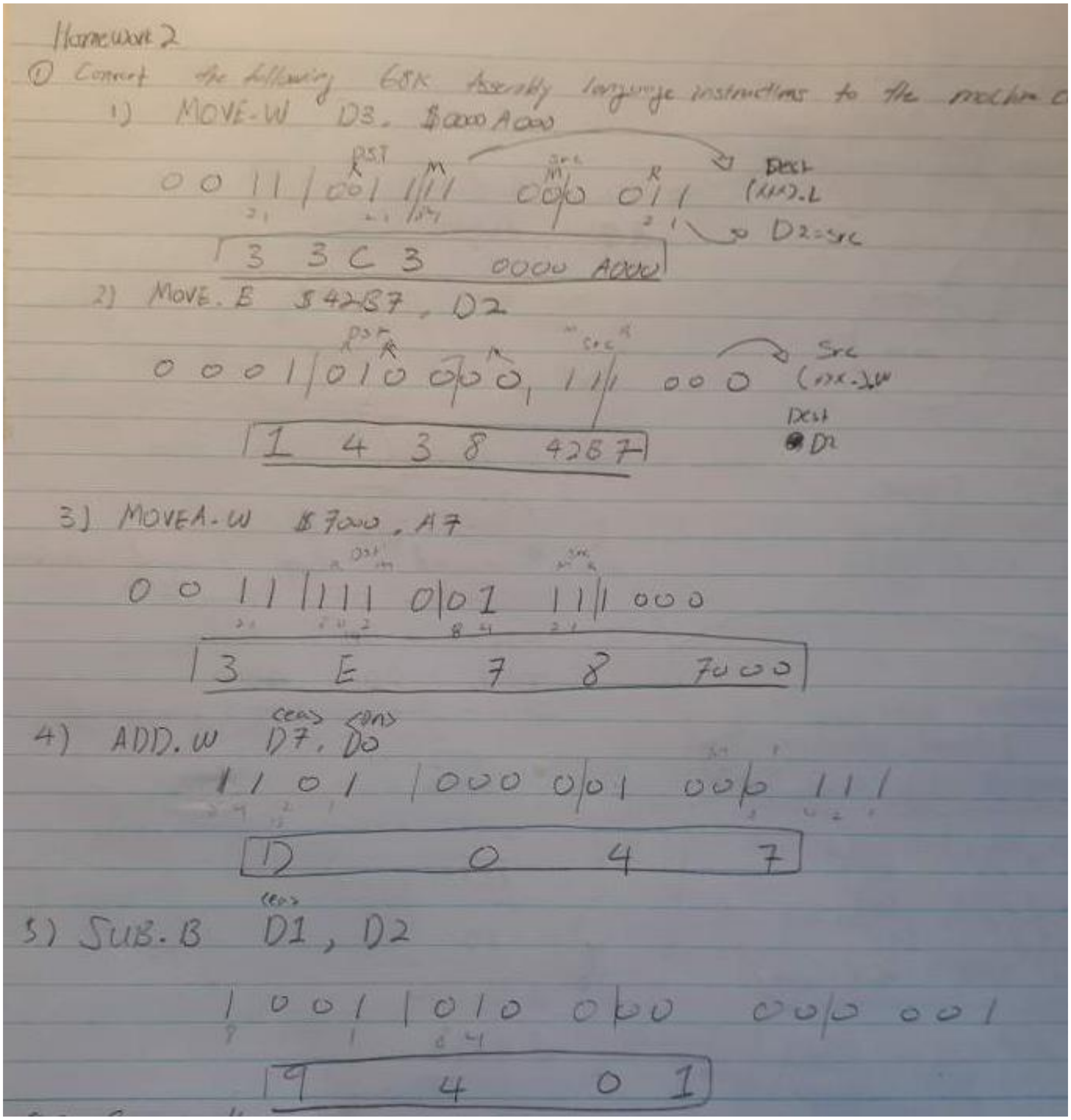
Jun Hyung Park

CSS 422

Homework 2

Q1. (5pts) Assemble the codes

Convert the following 68K assembly language instructions to the machine codes.



Homework 2

① Convert the following 68K Assembly language instructions to the machine c

1) MOVE.W D3, $0000A000

```
0 0 1 1 | 0 0 1  1 | 1 1   0 0 | 0  0 1 1     Dest (xxx).L
                                              D2=src
[3  3 C 3   0000 A000]
```

2) MOVE.B $4287, D2

```
0 0 0 1 | 0 1 0  0 | 0 0, 1 1 | 1  0 0 0     Src (xxx).W
                                              Dest D2
[1  4  3  8   4287]
```

3) MOVEA.W $7000, A7

```
0 0 1 1 | 1 1 1  0 | 0 1  1 1 | 1 0 0 0
[3   E   7   8   7000]
```

4) ADD.W D7, D0

```
1 1 0 1 | 0 0 0 0 | 0 1  0 0 | 0 1 1 1
[D   0   4   7]
```

5) SUB.B D1, D2

```
1 0 0 1 | 0 1 0  0 | 0 0   0 0 | 0 0 0 1
[9   4   0   1]
```

Q2. (10pts) Disassemble the codes

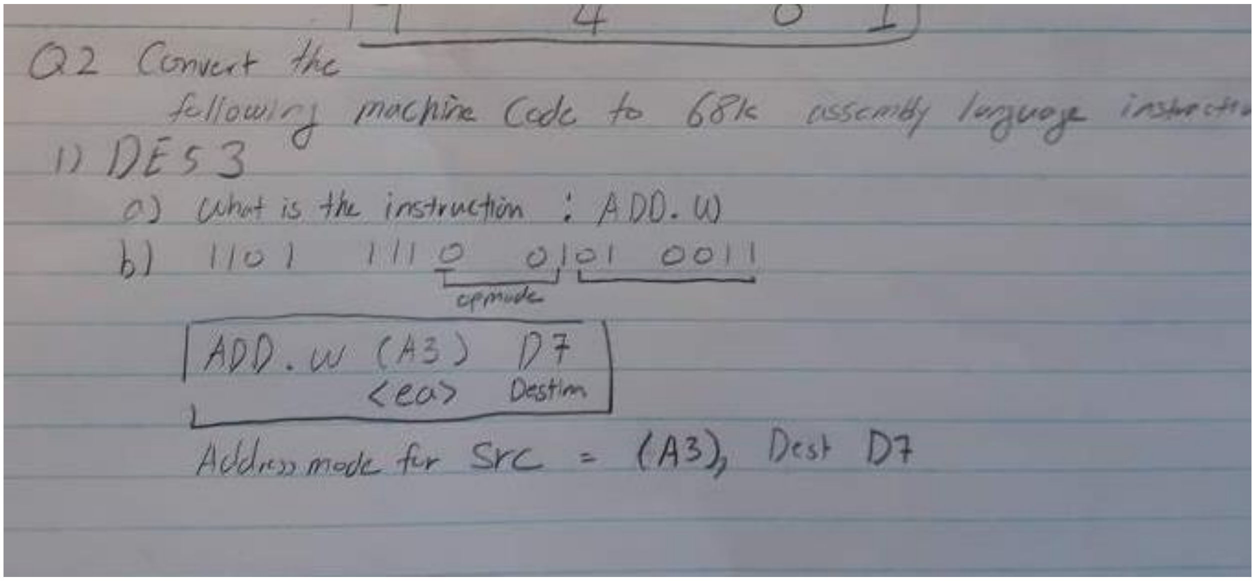
Convert the following machine codes to 68K assembly language instructions. Refer 68K manual, and only refer the MOVE, MOVEA, ADD, SUB instructions. Note that some immediate data format can be various.

For each question, you have to answer the followings and show your work. Without these steps, you will get zero

a) What is the instruction?

b) What is the addressing mode for source and destination?

Hint: Convert it to binary. Determine the instruction with opcode, then find the bits for source and destination. From the mode table in the manual, determine its addressing mode.



4 0 1

Q2 Convert the following machine Code to 68k assembly language instructi

1) DE53

a) What is the instruction : ADD. W

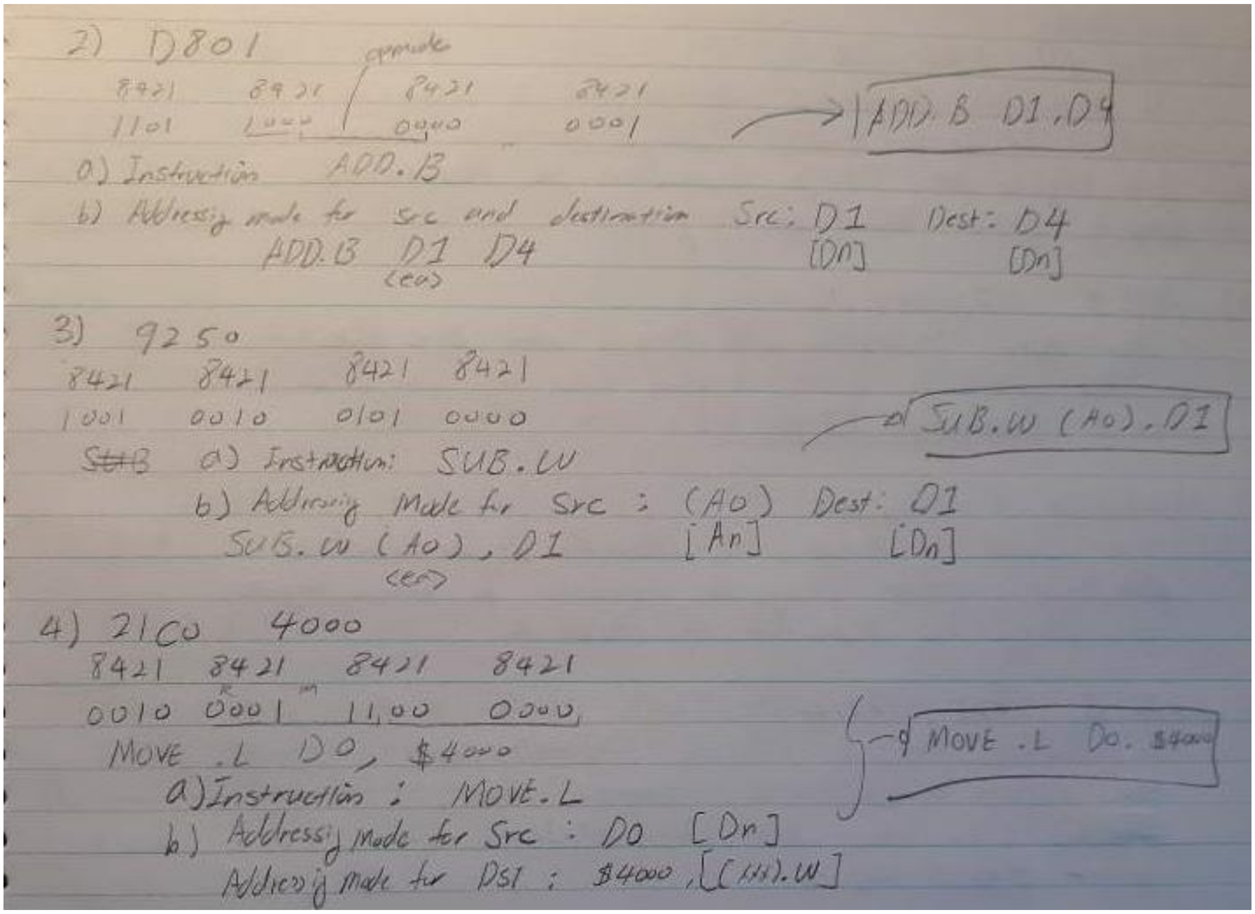
b) 1101 1110 0101 0011

opmode

ADD.w (A3) D7

<ea> Destin

Address mode for Src = (A3), Dest D7



2) D801

opmode

8421 8421 8421 8421
1101 1000 0000 0001

→ ADD.B D1,D4

a) Instruction ADD.B

b) Addressing mode for src and destination Src: D1 Dest: D4

ADD.B D1 D4 [Dn] [Dn]

<ea>

3) 9250

8421 8421 8421 8421
1001 0010 0101 0000

→ SUB.W (A0).D1

~~SUB~~ a) Instruction: SUB.W

b) Addressing Mode for Src : (A0) Dest: D1

SUB.W (A0), D1 [An] [Dn]

<ea>

4) 21C0 4000

8421 8421 8421 8421
0010 0001 11,00 0000

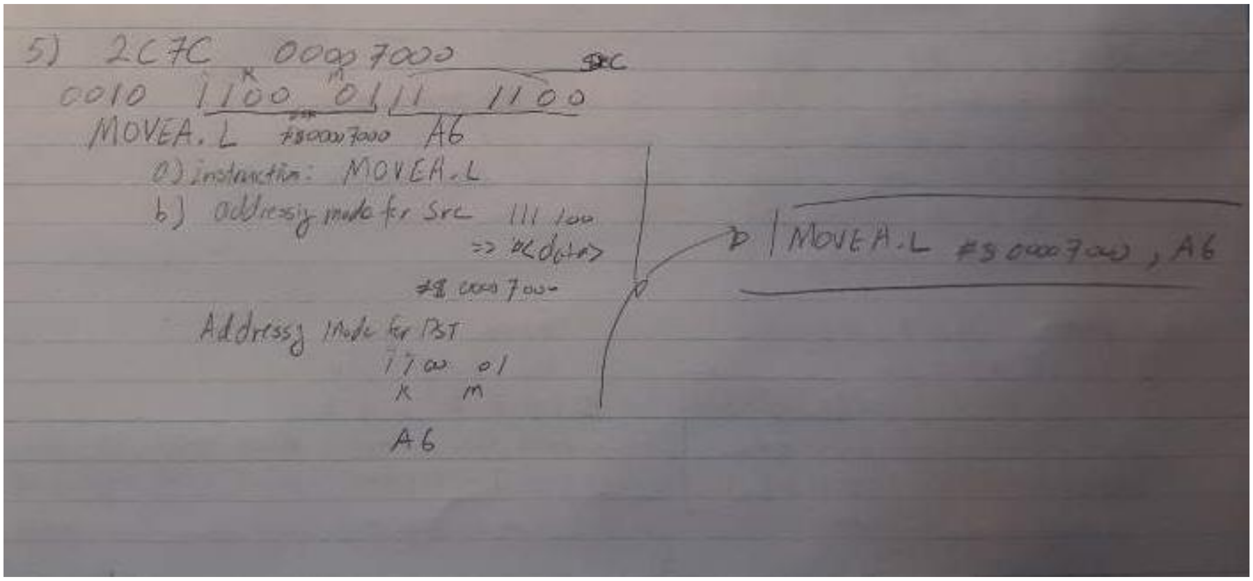
MOVE .L D0, $4000

→ MOVE .L D0. $4000

a) Instruction : MOVE.L

b) Addressing mode for Src : D0 [Dn]

Addressing mode for DST : $4000 ,[(xxx).W]



5) 2C7C 00007000

SRC

0010 1100 0111 1100

MOVEA.L #$00007000 A6

a) instruction: MOVEA.L

b) addressing mode for Src 111 100

=> #<data>

#$0000700

→ MOVEA.L #$0000700 , A6

Addressing Mode for DST

1100 01

k m

A6

Q3. (5 pts) Error Finding

Each of the following 68K assembly language instructions will cause an assembler error. Examine each instruction and explain how to fix it.

1) MOVE.B $A000, A3

The Destination cannot be address register if the instruction is MOVE. It has to be either change Destination Register to Data register or Change the instruction to MOVEA and size to either word or long word.

2) ADD.B #$1000, D2

The source immediate data size is not byte size. So, either change the immediate data size of change the size of instruction like ADD.W

3) MOVEA.W $1234, D0

Since instruction is MOVEA, the destination register cannot be data register. Either change the destination to address register or change the instruction to MOVE.W

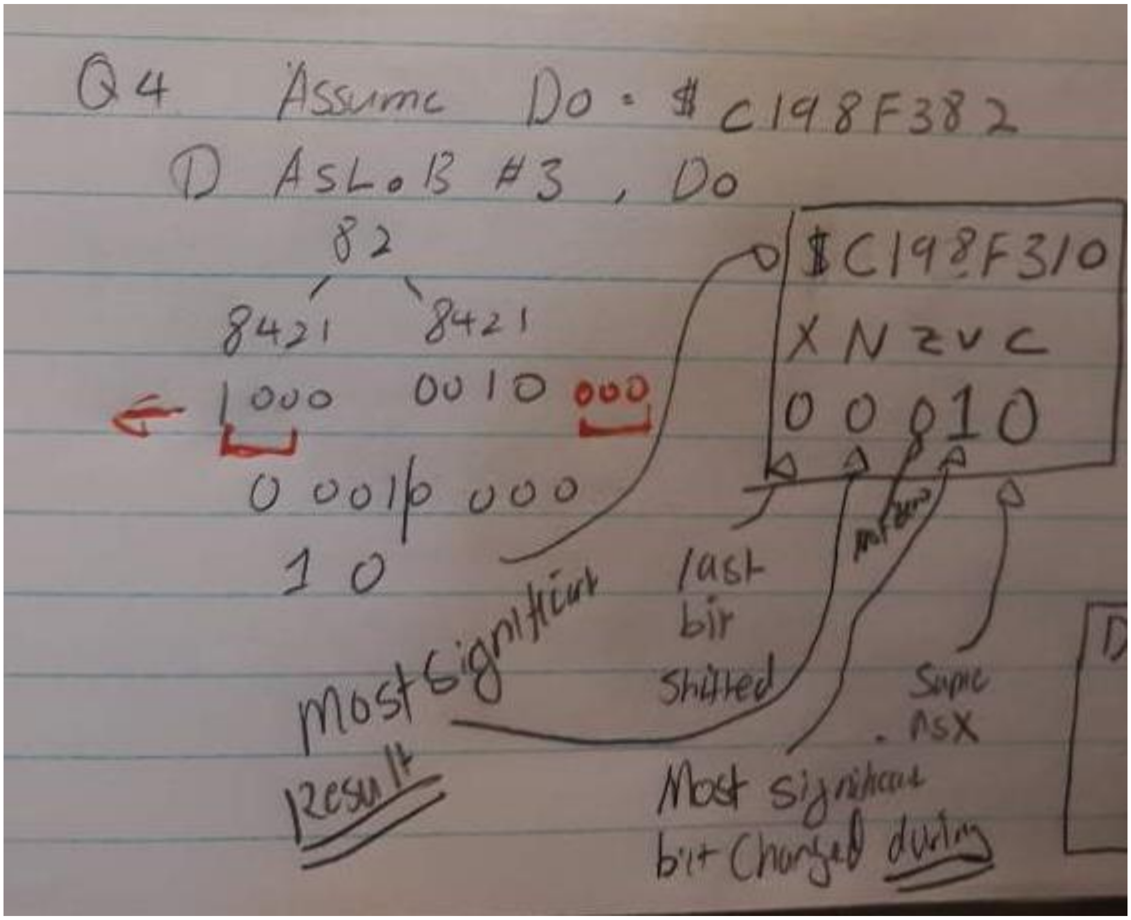
4) ANDI.B #23, #$100

The instruction ANDI.B requires #<data>, <ea> and effective address cannot be immediate data for ANDI.B. Furthermore, the destination cannot be immediate data. Therefore, either use Dn, (An), (An)+, etc… except An and #<data>.

5) SUBI.B D3, %1000

The source D3 is the problem. Because the syntax of instruction SUBI is SUBI #<data>, <ea> and D3 is not immediate data, it is data register. So, to fix it, simply change D3 to immediate data whatever you want but size of Byte.

Q4. (5 pts) For each of the operations below, assume that D0 contains the value $C198F382 and initially XNZVC=00000. Then, evaluate the value in D0 and the state of the CCR.


Q4 Assume D0 = $C198F382

① ASL.B #3, D0

Therefore, D0: C198F310 and 0 0 0 1 0 for XNZVC bits.

I think it is kind of hard to see why XNZVC is like that, so I will write it in here.

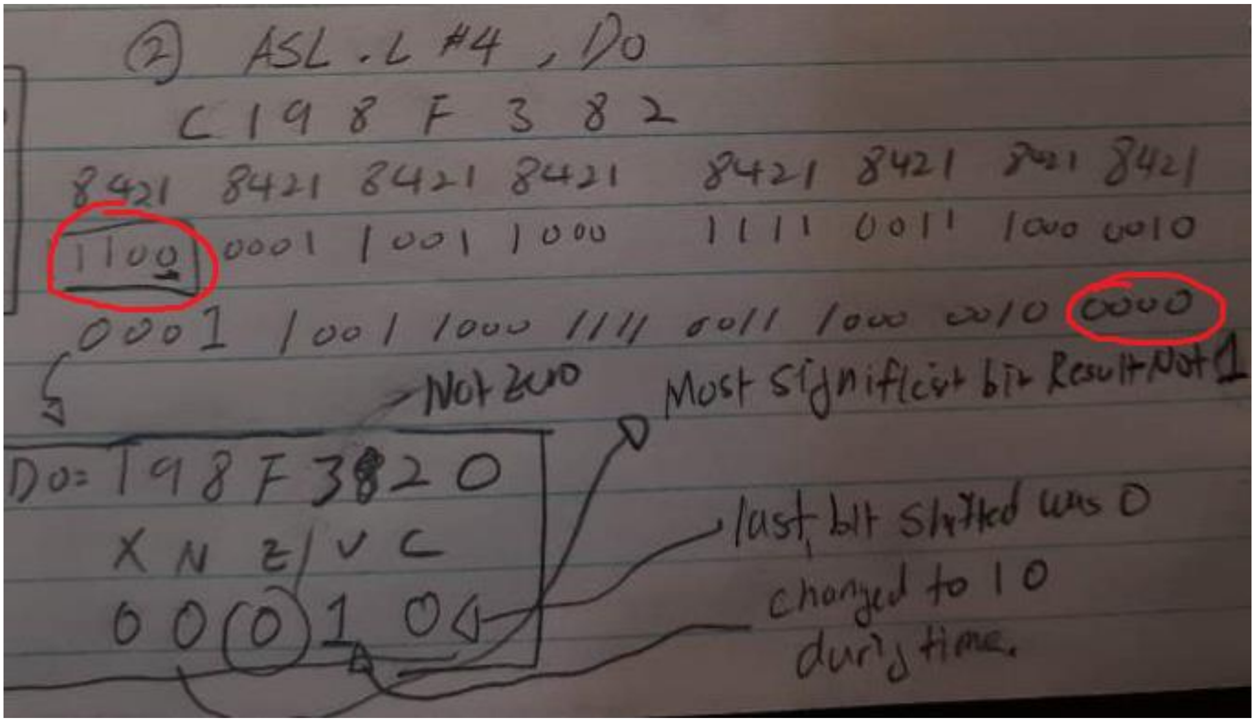
X: 0 because the last bit shifted out was 0

N: 0 because most sig fig on the result is 0

Z: 0 because its not 0

V: 1 because most significant bit changed over time during shift

C: 0 because last bit shifted out was 0



② ASL.L #4, D0

C 1 9 8 F 3 8 2

8421 8421 8421 8421 8421 8421 8421 8421

1100 0001 1001 1000 1111 0011 1000 0010

0001 1001 1000 1111 0011 1000 0010 0000

Not Zero

Most significant bit Result Not 1

D0= 198F3820

X N Z V C

0 0 0 1 0

last bit shifted was 0

changed to 1 0 during time.

D0: 198F3820 and 0 0 0 1 0 for XNZVC

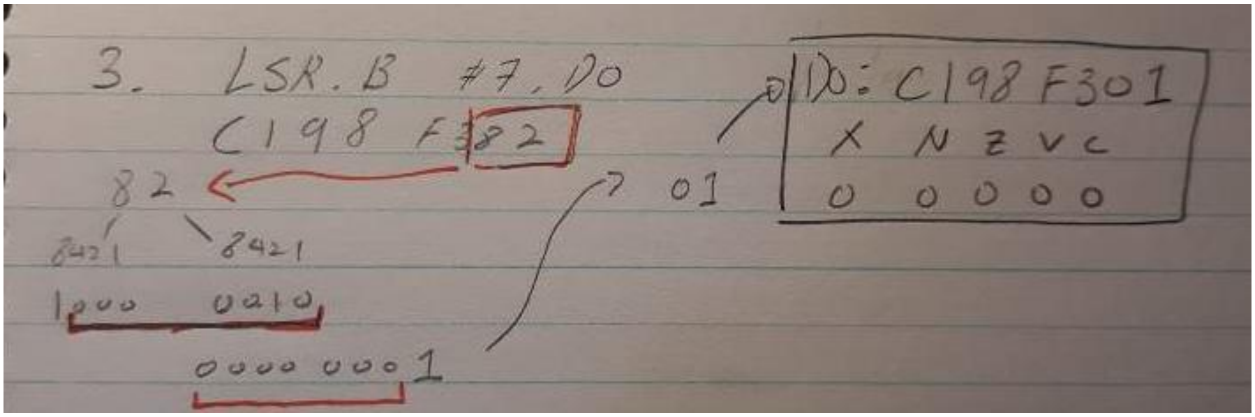
X: 0 because the last bit shifts out was 0

N: 0 because most significant bit is 0

Z: 0 because not 0

V: 1 because most significant bits has changed during shift.

C: 0 because last bit shifts out was 0



3. LSR.B #7, D0

C 1 9 8 F 3 8 2

82

8421 8421

1000 0010

0000 0001

01

D0: C198F301

X N Z V C

0 0 0 0 0

D0: C198F301 and 0 0 0 0 0 for XNZVC

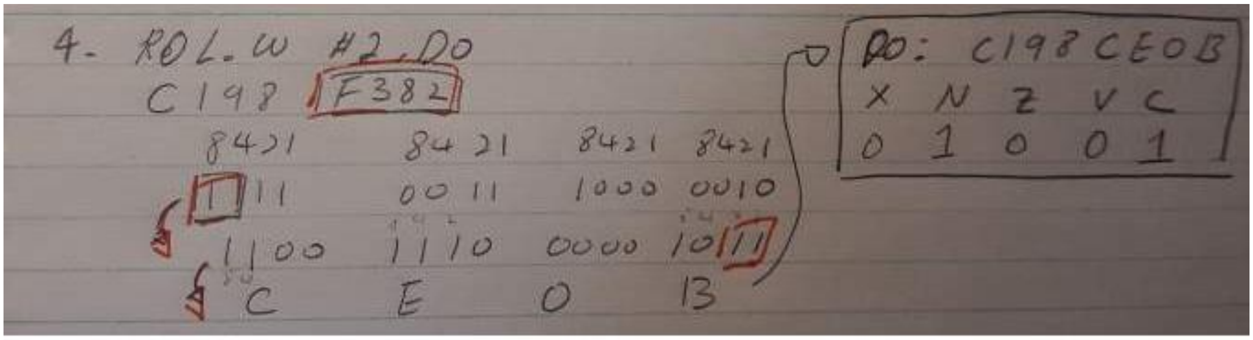
X: 0 because last bit shifted out is 0

N: 0 because the result is not negative

Z: 0 because it is not 0

V: Always clear so 0 because we started as 0

C: 0 because the last bit shifted out is 0



4. ROL.W #2,D0

C198 F382

```
8421   8421   8421   8421
1111   0011   1000   0010
1100   1110   0000   1011
 C      E      0      B
```

D0: C198CE0B

X N Z V C

0 1 0 0 1

D0: C198CE0B and 0 1 0 0 1 for XNZVC

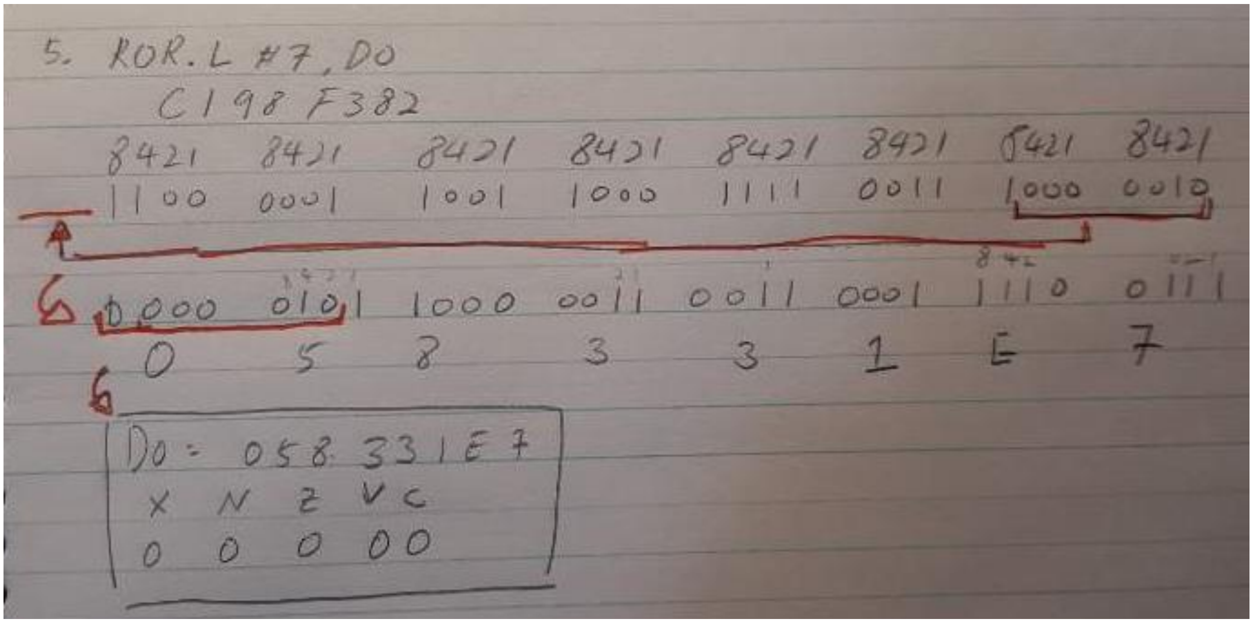
X: Not affected so just 0 (started as 0)

N: 1 because the most significant bit of the result is 1

Z: 0 because it is not zero

V: Always clear so 0 (started as 0)

C: 1 because the last bit rotated out was 1



5. ROR.L #7,D0

C198 F382

```
8421  8421  8421  8421  8421  8421  8421  8421
1100  0001  1001  1000  1111  0011  1000  0010
0000  0101  1000  0011  0011  0001  1110  0111
 0     5     8     3     3     1     E     7
```

D0 = 058331E7

X N Z V C

0 0 0 0 0

D0: 058331E7 and 0 0 0 0 0 for XNZVC

X: Not affected so just 0

N: 0 because the most significant bit of the result is 0

Z: 0 because it is not zero

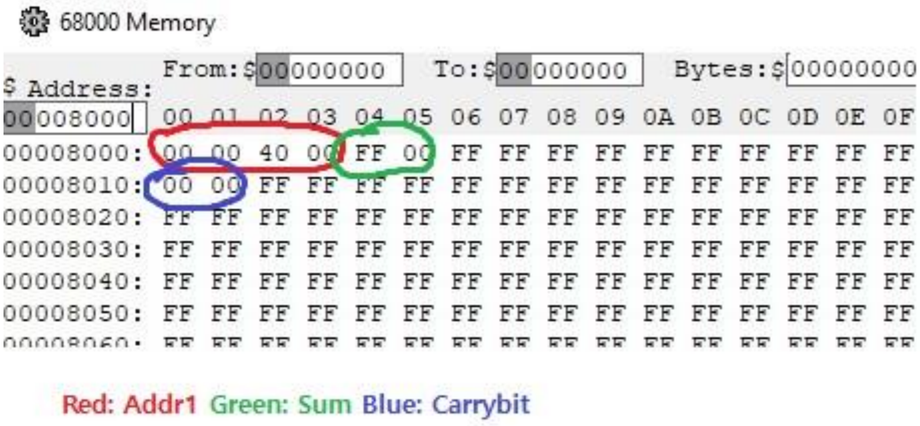
V: always clear (0 because we started as 0).

C: 0 because the last bit rotated out was 0.

Q5. (5 pts) Pattern Finding and Cumulative program.

Test 1: Nothing Match

Memory of Addr1, Addsum, CarryBit

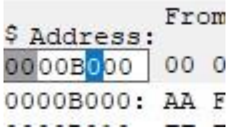


68000 Memory

```
$ Address:  From:$00000000   To:$00000000   Bytes:$00000000
00008000    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00008000:   00 00 40 00 FF 00 FF FF FF FF FF FF FF FF FF FF
00008010:   00 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00008020:   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00008030:   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00008040:   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00008050:   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00008060:   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

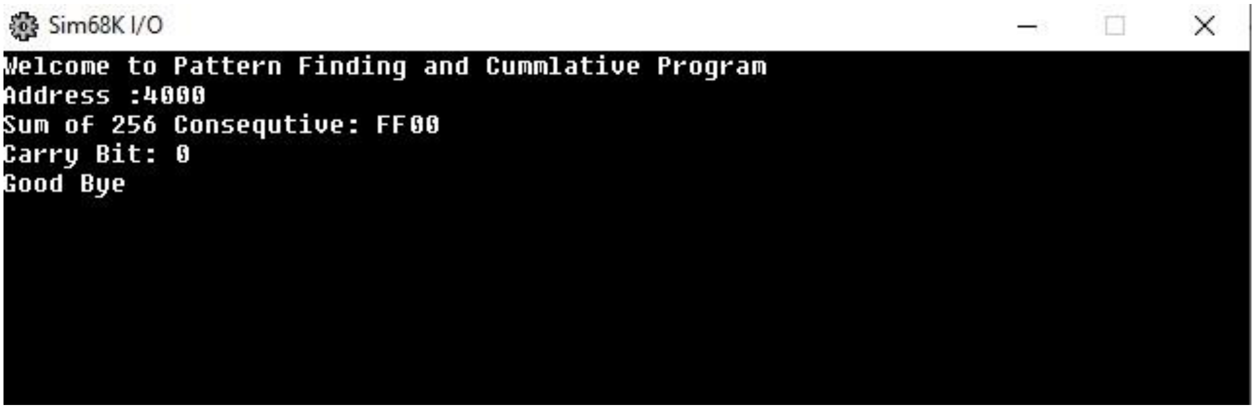
Red: Addr1 Green: Sum Blue: Carrybit

Target (B000)



```
$ Address:  From
0000B000    00 0
0000B000:   AA F
```

Output

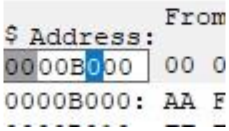


Sim68K I/O

```
Welcome to Pattern Finding and Cummlative Program
Address :4000
Sum of 256 Consequtive: FF00
Carry Bit: 0
Good Bye
```

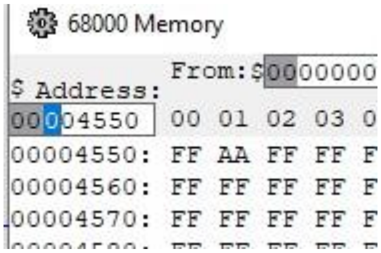
Test 2: Found Target At $4551

Target



```
                From
$ Address:
0000B000     00 0
0000B000: AA F
```

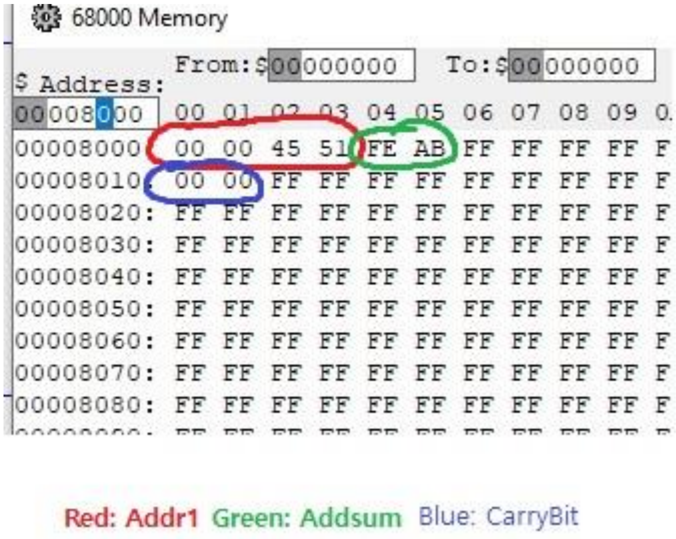
AA had been in memory location 4551



68000 Memory

```
             From:$0000000
$ Address:
00004550     00 01 02 03 0
00004550: FF AA FF FF F
00004560: FF FF FF FF F
00004570: FF FF FF FF F
```

Memory of Addr1, Addsum, CarryBit

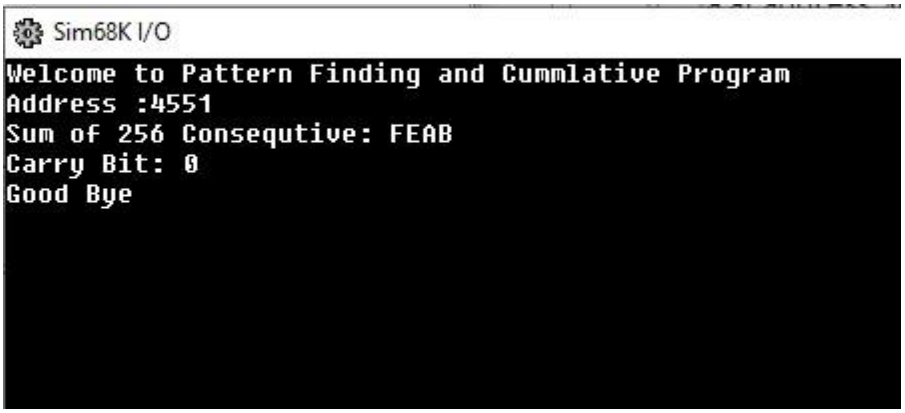


68000 Memory

```
             From:$00000000   To:$00000000
$ Address:
00008000     00 01 02 03 04 05 06 07 08 09 0.
00008000:  00 00 45 51 FE AB FF FF FF FF F
00008010:  00 00 FF FF FF FF FF FF FF FF F
00008020:  FF FF FF FF FF FF FF FF FF FF F
00008030:  FF FF FF FF FF FF FF FF FF FF F
00008040:  FF FF FF FF FF FF FF FF FF FF F
00008050:  FF FF FF FF FF FF FF FF FF FF F
00008060:  FF FF FF FF FF FF FF FF FF FF F
00008070:  FF FF FF FF FF FF FF FF FF FF F
00008080:  FF FF FF FF FF FF FF FF FF FF F
```

Red: Addr1 Green: Addsum Blue: CarryBit

Result



Sim68K I/O

```
Welcome to Pattern Finding and Cummlative Program
Address :4551
Sum of 256 Consequtive: FEAB
Carry Bit: 0
Good Bye
```