1. (a) \[7 \times 16^3 + 15 \times 16^2 + 3 = 32515\]
(b) \[\text{A3FA} = 1010001111010\]
(c) Using right-to-left division as on p. 753 of Bobrow, and remainder first reminder is $15 = \text{F}$ in hexadecimal notation.

\[
\begin{array}{c|cccc}
  & 15 & 15 & \div & 16 \\
\hline
  & 255 & 16 & \downarrow & 255 = \text{FF} \\
& 16 & 1 & & \\
\end{array}
\]


Check: \[256 - 16^2 = 100\]
\[255 = 256 - 1 = 100 - 1 = -\text{FF}.
\]

(d) -12F:
Start with 12F, then complement and add 1 to get negative number.

\[
\begin{array}{c|cccc}
  & 7 & \downarrow & \div & 16 \\
\hline
  & 127 & 16 & \downarrow & 127 = \text{7F} \\
& 16 & 1 & & \\
\end{array}
\]

In 16 bits: \[\text{7F} = \text{007F}.
\]
Complement: complement of 0 is \[7 (0111) \] is 8 (1000)
Complement of \[\text{007F} \] is \[\text{FF80} \]
Add 1: \[-\text{127} = \text{FF81} \]

Check: \[\text{FF81} = \text{0000000111111100} \]
Find positive magnitude:
\[\text{000000000000001} + 1 = \text{111111} \]
\[= \text{7F} = 127 \] \quad (OK).

2. \[\text{F03F} \] is negative since left-most bit is set.
Complement and add 1 to get positive magnitude:
\[C = 12 = \text{1100} \] complement is 0011 = 3
Magnitude: \[\text{F03F} + 1 = \text{F2FD} \]
\[= 3 \times 256 + 15 \times 16 + 13 = 1021 \]
So \[\text{F03F} = -1021 \]
2. (a) jmp causes the program to transfer to the specified address. jsr also transfers to the specified address but also "pushes" the address of the next instruction onto the run-time stack. Both instructions would cause the program to go to line 29 but the rts command in the subroutine would not work properly with the jmp. rts gets the "return address" for the subroutine from the stack to allow the program to continue on line 15 after completing the subroutine. If jmp were used, the rts command would not access the correct address.

(b) i) \[ sp \rightarrow sp-4 \equiv \$1000 - 4 = \$FFFF \]

ii) \$FFFF: \$1028 (the contents of the address in the stack ptr. is the location of the next instruction after the jsr instruction).

(c) Immediate addressing stores constants in instructions themselves. An example:
on line 21 we have move.w \#31, d0.
\# denotes immediate addressing.
The instruction puts \(31 = \#1E\) in d0.
(The constant is in the "code" area.)
\[(d) \quad \$40 = 01000000 \quad \{ \text{operation \"ors\" \$40 with \$20 bit by bit}\}
\quad \$20 = 00100000 \quad \Rightarrow \quad 01100000 = \$60. \quad \$60 = 1 \quad (\text{single quote})\]

(c) Instead of setting bit 5, we clear it, then look in range \$41-$46. Then subtract \$37 to get proper value \((\$41 - \$37 = \$4)\).

Changes:
41: Skip; and. b \#80F, d\#; clear bit 5
42: cmp. b \#841, d\#
44: cmp. b \#846, d\#
46: sub. b \#37, d\#; get proper value

Example: on line 41: d\#: Xxxxxx Xxxxxx \(\text{bitwise \"and\"}
\quad \$80F: 10111111
\quad \rightarrow \quad x\# \text{xor x\#}\#\#\#
\quad \text{result differs only in having bit 5 cleared (set to 0).}\)

3. (a) i) A5*
   ii) 50: npts is in the data area, would be accessed indirectly using the contents of A5.
   47, 66 do not use data area
   (A7 does not use data area although it does use indirect addressing relative to PC.)

(b) npts = 10, npts - 1 = 9 is placed in d\#3 on line 53 since the clear instruction (line 47) terminates the loop when d\#3 goes negative.
   If we had started with 10 we would have performed 11 iterations.

(c) i) \$FF is immediate addressing. The number \$FF is part of the instruction in the code area.
   ii) lower 8 bits of d\#: 11111111 (= -1)
iii) iv: Look up moveb instruction in manual excerpts. One finds the operation sets the N condition code bit if the result is negative and sets the Z bit if the result is 0. (Cleared otherwise). We saw the byte was negative (-1, in fact) since the sign bit is set. Thus, N=1, Z=0.

(d) moveb (a3)+, dφ

(e) These are data bytes (8 bits). The largest number one can store in 8 bits is 

\[ 2^8 - 1 = 256 - 1 = 255 \]

(assuming positive integers, which is what we want for the DAC). Thus the values must be less than 256.

(f) i) On line 29, dφ was set to #D_write = $580001.

ii) This is the address of a data register used to transfer data to the SCSI data lines as output (i.e., to set them high or low according to the bits sent). So the number in dφ is transferred to that register rather than to RAM.