Array For Loop Example:

```c
unsigned short a[10];
for(j=0; j<10; j++)
{
    if(a[j]==1) PORTT=0x04;
    else PORTT=0x00;
}
```

Programming Steps:
1. Initialize J
2. Compare J to 10
3. If Not Less than 10, End Loop
4. Else
   1. Load a[j]
   2. If a[j] == 1
      1. PORT T = 4
   3. Else
      1. PORT T = 0
   4. Increment J
   5. Repeat Loop (Step 2)

Assembly Code:
```assembly
ldaa #0     ; Initialize J
ldx #$3800  ; Initialize index to A[0]
Loop:
    cmpa #10    ; Compare J to 10
    bge EndLoop ; Else !(J<10)
    staa $3814  ; Store J to RAM
    ldd 0,X     ; Load A[J]
    cpd #1      ; Compare J to 1
    bne Else    ; Else !(A[J]==1)
    ldab #4     ; Value to write to PORT T
    bra EndIf
Else:
    ldab #0     ; Value to write to PORT T
    bra EndLoop
EndIf:
    stab PTT    ; Write value to PORT T
    adda #1     ; Increment J
    inx          ; Increment A[J]
    inx          ; Need to increment by 2
    bra Loop    ; Repeat Loop
EndLoop:
    ; do something else
```

Will this code work?
NO! ldd overwrite value of J
How can we correct this?
We need to store value of J to memory and reload it when needed.
There are 10 types of people in the world: Those who get binary and those who don’t.

Information vs. Data

- **Information**
  - An abstract description of facts, processes or perceptions
  - How can we represent information?
  - How can we represent changing information?
  - We need to associate different values with different events

- **Data**
  - Individual fact value or set of facts or values
  - Measurement or storage

### Data Representation

- The same data can be represented with:
  - Different symbols
    - English, Cyrillic, Arabic
  - Different numeric bases
    - Binary, Octal, Hexadecimal, Decimal
  - Different formats
    - Little Endian, Big Endian, Binary Coded Decimal

<table>
<thead>
<tr>
<th>International Talk Like a Pirate Day:</th>
<th>September 19th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date of your first ECE 372 midterm:</td>
<td>October 17, 2006</td>
</tr>
<tr>
<td>How much Bill Gates earns per second:</td>
<td>$250</td>
</tr>
<tr>
<td>Number of pennies to fill Empire State Building</td>
<td>1.8 Trillion</td>
</tr>
</tbody>
</table>
### Decimal Numbers
- Uses the **ten** numbers from 0 to 9
- Each column represents a power of 10

<table>
<thead>
<tr>
<th>Thousands (10⁰) column</th>
<th>Hundreds (10⁰) column</th>
<th>Tens (10¹) column</th>
<th>Ones (10²) column</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

\[1999_{10} = 1 \times 10^3 + 9 \times 10^2 + 9 \times 10^1 + 9 \times 10^0\]

### Binary Numbers
- Uses the **two** numbers from 0 to 1
- Every column represents a power of 2

<table>
<thead>
<tr>
<th>Eights (2⁰) column</th>
<th>Fours (2¹) column</th>
<th>Twos (2²) column</th>
<th>Ones (2³) column</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[1001_{2} = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0\]

\[= 8 + 0 + 0 + 1 = 9\]

### Convert the following value from binary (zero’s and one’s) to a decimal value:

\[100110_{2}\]

Choose your answer:
A) 100,110
B) 22
C) 38
D) 42
Converting from Decimal to Binary:
- Divide decimal number by 2 and insert remainder into new binary number.
- Continue dividing quotient by 2 until the quotient is 0.
- Example: Convert decimal number 12 to binary

<table>
<thead>
<tr>
<th>Decimal Number</th>
<th>Binary Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Example: Convert decimal number 12 to binary (cont.)

<table>
<thead>
<tr>
<th>Decimal Number</th>
<th>Binary Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Since quotient is 0, we can conclude that 12 is 1100 in binary.

Convert the following decimal value to a binary (zero's and one's) value:

54

Choose your answer:
A) 110110
B) 100010
C) 1000010
D) 10100100

Generally, a number can be converted from one base to another by
- Converting the number to base 10
- Then, converting the base ten number to the desired base using the divide-by-n method
- May not always be the easiest way...
Hexadecimal Numbers
- Uses the ten numbers from 0 to 9 and six letter A to F
- Each column represents a power of 16

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>101</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>110</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>111</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
<td>B</td>
</tr>
<tr>
<td>1100</td>
<td>12</td>
<td>C</td>
</tr>
<tr>
<td>1101</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
<td>E</td>
</tr>
<tr>
<td>1111</td>
<td>15</td>
<td>F</td>
</tr>
<tr>
<td>10000</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>10001</td>
<td>17</td>
<td>11</td>
</tr>
</tbody>
</table>

$000000_{16} = 16^5 + 5*16^2 + C*16^1 + A*16^0 = 41,157_{10}$

Hexadecimal Numbers
- Each position actually represents four base two positions
- Used as compact means to write binary numbers
- Often called just hex

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>101</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>110</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>111</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
<td>B</td>
</tr>
<tr>
<td>1100</td>
<td>12</td>
<td>C</td>
</tr>
<tr>
<td>1101</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
<td>E</td>
</tr>
<tr>
<td>1111</td>
<td>15</td>
<td>F</td>
</tr>
<tr>
<td>10000</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>10001</td>
<td>17</td>
<td>11</td>
</tr>
</tbody>
</table>

Convert the following hexadecimal value to a binary (zero’s and one’s) value:

$CAB_{16}$

Choose your answer:
A) 110111101010
B) 110001011001
C) 110010101011
D) 110010111010
ECE 372 – Microcontroller Design
Big vs. Little Endian

- Big Endian
  - MSB (Most Significant Byte) is at lowest address
  - 68000, MIPS, Sparc, HC12

- Little Endian
  - LSB (Least Significant Byte is at lowest address
  - 80x86, DEC

$FFFF lower address → MSB
$FFFF higher address → LSB

16 bit

ECE 372 – Microcontroller Design
Positive vs. Negative Numbers

- Negative numbers are common
  - How can we represent negative numbers in binary?
    - Signed-magnitude
      - Use leftmost bit for sign bit
        - So -5 would be: 1101
    - One’s Complement
      - Invert all bits for negative numbers
        - So -5 would be: 1010

- Two’s Complement
  - Allows us to perform subtraction using addition
    - No need for dedicated subtractor within CPU’s ALU
  - Two’s complement of a number added to the number itself will equal zero
    - So -5 would be: 1011
    - $1011_2 + 0101_2 = 0000_2$ (with carry of 1, ignored)
  - Invert all bits and add 1 to get complement of a number
    - Fast conversion: find first 1 from right, invert after 1

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Signed-Magnitude</th>
<th>One’s Complement</th>
<th>Two’s Complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>+4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>+3</td>
<td>011</td>
<td>011</td>
<td>011</td>
</tr>
<tr>
<td>+2</td>
<td>010</td>
<td>010</td>
<td>010</td>
</tr>
<tr>
<td>+1</td>
<td>001</td>
<td>001</td>
<td>001</td>
</tr>
<tr>
<td>0</td>
<td>000</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>-1</td>
<td>101</td>
<td>110</td>
<td>111</td>
</tr>
<tr>
<td>-2</td>
<td>110</td>
<td>101</td>
<td>110</td>
</tr>
<tr>
<td>-3</td>
<td>111</td>
<td>100</td>
<td>101</td>
</tr>
<tr>
<td>-4</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
</tbody>
</table>

Are we missing a value?
Positive vs. Negative Numbers

- **Data Ranges**
  - **Unsigned**: 0 to $2^n - 1$
  - **Signed-Magnitude**: $-2^{n-1}$ to $2^{n-1} - 1$
  - **One’s Complement**: $-2^{n-1} + 1$ to $2^{n-1} - 1$
  - **Two’s Complement**: $-2^{n-1}$ to $2^{n-1} - 1$

Detecting the two’s complement representation for the following decimal numbers (assume we are using 8-bit binary numbers):
- -1
- -11
- -15
- 22
- -101

Overflow

- Occurs when a result cannot be represented with given number of bits
- Either the result is too large magnitude of positive or negative

Detecting Overflow

- Can detect overflow by detecting when the two numbers’ sign bits are the same but the result’s sign bit is different
- If the two numbers’ sign bits are different, overflow is impossible
  - Adding a positive and negative can’t exceed largest magnitude positive or negative

<table>
<thead>
<tr>
<th>Addition</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1 + 1</td>
<td>0</td>
</tr>
<tr>
<td>-11 + 1</td>
<td>0</td>
</tr>
<tr>
<td>-15 + 22</td>
<td>0</td>
</tr>
<tr>
<td>-101 + 22</td>
<td>0</td>
</tr>
</tbody>
</table>

-000  overflow 
-001  overflow
-010  no overflow
-011  no overflow
-100  no overflow
-101  no overflow
-110  no overflow
-111  no overflow
ECE 372 – Microcontroller Design

Overflow

- Binary Coded Decimal
  - Each digit of a decimal number is represented as a 4-bit binary number
  - Often used for 7-segment displays

<table>
<thead>
<tr>
<th>Binary</th>
<th>BCD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>Undefined</td>
<td>10</td>
</tr>
<tr>
<td>Undefined</td>
<td>11</td>
</tr>
<tr>
<td>Undefined</td>
<td>12</td>
</tr>
<tr>
<td>Undefined</td>
<td>13</td>
</tr>
<tr>
<td>Undefined</td>
<td>14</td>
</tr>
<tr>
<td>Undefined</td>
<td>15</td>
</tr>
</tbody>
</table>

Real Numbers

- How can we represent a real number (i.e. numbers that contain a fractional part)?
  - Fixed Point Numbers
  - Floating Point Numbers

Note: our C compiler already has built-in routines to deal with real numbers
  - However the computational needs will be significant.
Fixed Point Numbers
- Real number with fixed number of digits before and after radix point
  - N-bits used to represent integer part
  - M-bits used to represent fractional part
  - Unsigned range: 0 to $2^N - 1/2^M$

$0.0100 = 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0 \cdot 2^{-3} + 0 \cdot 2^{-4}$

$0.1011 = 0 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 0 \cdot 1 + 1 \cdot 0.5 + 1 \cdot 0.25 + 0 \cdot 0.125 + 0 \cdot 0.0625$

$= 4.25$

$1.1111 = 1 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 + 1 \cdot 0.5 + 1 \cdot 0.25 + 0 \cdot 0.125 + 1 \cdot 0.0625$

$= 15.5625$

$0100.0100 + 1011.1001 = 1111.1101$

$1.1111 + 1.1111 = 1 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 + 1 \cdot 0.5 + 1 \cdot 0.25 + 1 \cdot 0.125 + 1 \cdot 0.0625$

$= 15.8125$
Floating Point Numbers

- Real number representation similar to scientific notation
  \[ x = M \times B^E \]

- Base (B)
  - Base of the numbering systems considered
  - Binary (2) for computer based implementations
  - We will assume base of 2 for remaining description

- Sign (S)
  - Indicating positive or negative number
  - 0 – Positive, 1 – Negative

- Mantissa (M)
  - Digits corresponding to the magnitude
  - Stored in a normalized form, where the first bit is assumed to be 1
  - 1 is the only possible non-zero number in binary
  - Remaining bits correspond to fraction values (similar to fixed point)

- Exponent (E)
  - Needs to represent both positive and negative values
  - Stored exponent is adjusted using the exponent bias
    \[ E_{\text{stored}} = E_{\text{actual}} - E_{\text{bias}} \]
  - Example, 8-bit exponent:
    \[ E_{\text{bias}} = 2^8 - 1 = 127 \]
    \[ E_{\text{actual}} = 10000000_2 = 128 \]
    \[ E_{\text{stored}} = 128 - 127 = 1 \]

Example: Convert the value -118.625 to floating point representation

1. Determine sign bit:
   - -118.625 is negative, S = 1
2. Convert to binary:
   \[ 118.625 = 1110110.101_2 \]
3. Normalize number
   \[ 1.110110101 \times 2^6 \]
4. Determine exponent
   \[ E_{\text{actual}} = 6 + 127 = 133 \]
   \[ E_{\text{stored}} = E_{\text{actual}} + E_{\text{bias}} = 133 + 127 = 260 \]


\[ \begin{array}{ccc}
S & E & M \\
\hline
1 & 8 & 23 \\
\end{array} \]
Floating Point Numbers

- Real number representation similar to scientific notation
  - \( x = M \times 2^E \)

- Zero
  - Due to assuming a leading 1 in the mantissa, we cannot directly represent the value 0 using floating point
  - Defined special case for value of 0
  - Define special case: Exponent and mantissa of all 0's corresponds to the value 0

- Other special cases exist:
  - +/- Infinity
  - Demoralized value
  - Not a Number

IEEE Standard for Binary Floating-Point Arithmetic (IEEE 754)

- Single Precision Floating Point

\[
\begin{array}{c|c}
S & E & M \\
\hline
1 & 8 & 23 \\
\end{array}
\]

- Double Precision Floating Point

\[
\begin{array}{c|c}
S & E & M \\
\hline
1 & 11 & 52 \\
\end{array}
\]

What does the bit pattern mean: 1011 1001

- Unsigned: 185 decimal
- Sign-Magnitude: -57 decimal
- 1's complement: -70 decimal
- 2's complement: -71 decimal
- Fixed point 4bit.4bit: 11.5625 decimal
- BCD:
  - 10112 = B (Undefined)
  - 10012 = 910
- ASCII: '9'
- HC12 Opcode: $39 = \text{ADCA (Add with Carry to A, Ext. Addressing)}$
Digital Audio
- 44.2kHz sampling rate
- 16bit/channel $1..65536=2^{16}$
- Stereo
- How many seconds of sound fit in
  - 32kbyte EPROM mono?
  - 1.4 Mbyte floppy mono?
  - 660Mbyte CD Rom stereo?