Question 1:

Part A (10 points)
The data memory and register file contents are given below:

<table>
<thead>
<tr>
<th>Data Memory</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0800</td>
<td>W0 0x0800</td>
</tr>
<tr>
<td>0x0802</td>
<td>W1 0x0802</td>
</tr>
<tr>
<td>0x0804</td>
<td>W2 0x0806</td>
</tr>
<tr>
<td>0x0806</td>
<td></td>
</tr>
</tbody>
</table>

Indicate the address and the contents of the register/memory as the result of each instruction below:

1. add w0, w1, w2
   - 0x1122

2. add w0, [w1], [w2]
   - 0x3344
   - 0x0806

3. add [w0], [w1], w2
   - Error

4. subr w0, w1, w2
   - 0x5566
   - 0x0802

5. sub [w0], w1, [w2]
   - Error

Part B (10 points)
Indicate the contents of the data memory address, 0x0800 when the following PIC24 assembly program ends.

```assembly
mov #ABCD, w1
mov #1234, w2
sub w1, w2, w3
bra C, loc1
dec2 w3, w3
mov w3, 0x0800
loc1:
mov w3, 0x0800
```

0x0800 0x9987
Question 2:

Part A (15 points)
The contents of w0 and data memory address 0x0800 are shown as follows:

\[
\begin{array}{c|c|c}
   & 0x8000 & 0xEF11 \\
\hline
w0 & 0xABCD & 0x0800 \\
\end{array}
\]

Write PIC24 assembly code such that the following operation takes place and indicate the register and memory contents at 0x0800 after each instruction.

\[\text{mem [0x0800]} \ ^{\wedge} \ 0x0F0F \rightarrow \text{mem [0x0800]}\]

Part B (25 points)
Write a PIC24 assembly code without using move instruction to accomplish the following task and indicate the register and memory contents at 0x0800 after each instruction.

\[0x8122 \rightarrow \text{mem [0x0800]}\]

Question 3:

Part A (10 points)
The following C program is given:

\[
\begin{align*}
\text{unsigned int } & \ k, \ j, \ p, \ q; \\
& \ k = k + j; \\
& \ k = k + 1; \\
& \ p = p - q; \\
& \ p = p - 2;
\end{align*}
\]

Write a PIC24 assembly code for this program with all compiler directives.

Part B (30 points)
The data memory location 0x0800 has the following data:

\[
\begin{array}{c|c|c}
   & 0x8000 & 0x12 \\
\hline
\text{BH} & 0xBH & \text{BL} \\
\end{array}
\]

where, BH and BL correspond to high and low bytes.

Write a PIC24 assembly code such that the least (most) significant byte switches to the most (least) significant byte location to result in the following for 0x0800:

\[
\begin{array}{c|c|c}
   & 0x8000 & 0x12 \\
\hline
\text{BH} & 0x12 & \text{BL} \\
\end{array}
\]

Show the register/memory contents after every instruction in your code.
**Question 1:**

**Part A:**

<table>
<thead>
<tr>
<th>0x0800</th>
<th>0x1122</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0802</td>
<td>0x3344</td>
</tr>
<tr>
<td>0x0804</td>
<td>0x5566</td>
</tr>
<tr>
<td>0x0806</td>
<td>0x7788</td>
</tr>
<tr>
<td>w0</td>
<td>0x0800</td>
</tr>
<tr>
<td>w1</td>
<td>0x0802</td>
</tr>
<tr>
<td>w2</td>
<td>0x0806</td>
</tr>
</tbody>
</table>

```
add w0, w1, w2
add w0, [w1], [w2]
add [w0], [w1], [w2]
subr w0, w1, w2
subr [w0], w1, [w2]
```

**Part B:**

```
mov # 0x4BCD, w1
mov # 0x1234, w2
sub w1, w2, w3
bra C, loc1
dec2 w3, w3
mov w3, 0x0800
loc1:
mov w3, 0x0800
```

- w2: 0x1002
- 0x3844 0x0806
- Compiler error
- w2: 0x0002
- Compiler error

- w3: 0x9999
- c = 0
- w3: 0x9997
- 0x9997 0x0800 4
Question 2:

Part A:

\[ \text{Mov} \ #0\times0\text{FOF}, \ w0 \Rightarrow 0\times0\text{FOF} \rightarrow \text{Reg}[w0] \]

\[ \text{Xor} \ 0\times0\text{8000} \Rightarrow \text{Mem}[0\times0\text{8000}] \rightarrow \text{Reg}[w0] \]

\[ \therefore \text{Mem}[0\times0\text{8000}] \rightarrow \text{Mem}[0\times0\text{8000}] \]

Thus: \[ 0\times0\text{8000} \rightarrow 0\times0\text{E01E} \]

Part B:

To achieve \[ 0\times8\text{122} \rightarrow \text{Mem}[0\times0\text{8000}] \Rightarrow \]

\[ \text{clr} \ 0\times0\text{8000} \]

\[ \text{bsct} \ 0\times0\text{8000}, \#15 \]

\[ \text{bsct} \ 0\times0\text{8000}, \#8 \]

\[ \text{bsct} \ 0\times0\text{8000}, \#5 \]

\[ \text{bsct} \ 0\times0\text{8000}, \#1 \]
Question 3:

Part A:  

unsigned int k, j, p, q;

k = k + j;
k = k + 1;
p = p - q;
p = p - 2;

Thus:

k: *space 2
;
q: *space 2
mov j, wo
add k
inc k
mov q, wo
sub p
dec2 p

mem[j] → Reg[wo]
mem[k] + Reg[wo] → mem[k]
mem[k] + 1 → mem[k]
mem[q] → Reg[wo]
mem[p] - Reg[wo] → mem[p]
mem[p] - 2 → mem[p]
Part B:

\[ \text{mov}, b \ 0x0800, w0 \]
\[ \text{sl} \ w0, \#8, w0 \]
\[ \text{mov} \ 0x0800, w1 \]
\[ \text{lsr} \ w1, \#8, w1 \]
\[ \text{mov}, b \ w1, w0 \]
\[ \text{mov} \ w0, 0x0800 \]
Question 1 (20 points):

The following are the contents of the data memory and the register file (SFR).

<table>
<thead>
<tr>
<th>Data Memory</th>
<th>SFR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0800</td>
<td>W0 0x0800</td>
</tr>
<tr>
<td>0x0802</td>
<td>W1 0x0802</td>
</tr>
<tr>
<td>0x0804</td>
<td>W2 0x0804</td>
</tr>
</tbody>
</table>

Write the operation for each instruction in terms of equation(s), the destination address and the contents of the result (in data memory or SFR):

(1) mov.b [W0], W1  
   Equation: NOT POSSIBLE

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(2) mov [W0], [W1]  
   Equation: mem [0x0800] \rightarrow mem [0x0802]

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0802</td>
<td>0x1000</td>
</tr>
</tbody>
</table>

(3) add W2, [W1], [W0]  
   Equation: \text{Reg} [w2] + \text{mem} [\text{Reg} [w1]] \rightarrow \text{mem} [\text{Reg} [w0]] \text{ Reg}[w0] + \text{mem} [0x0802] \rightarrow \text{mem} [0x0800] \uparrow 0x2000

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0800</td>
<td>0x2804</td>
</tr>
</tbody>
</table>

(4) mov [++W0], W1  
   Equation: Reg [w0] + 2 = 0x0800 + 2 = 0x0802 \text{ Reg}[w0] + \text{mem} [0x0802] = 0x2000 \rightarrow \text{Reg} [w1]

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>0x2000</td>
</tr>
</tbody>
</table>

(5) mov [W0++], W1  
   Equation: \text{mem} [\text{Reg} [w0]] = \text{mem} [0x0800] \rightarrow \text{Reg} [w1] \text{ Reg}[w0] + 2 = 0x0802 \rightarrow \text{Reg} [w0]

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>0x1000</td>
</tr>
</tbody>
</table>
(6) sub W2, [W0], [W0]  
Equation: \[ \text{Reg}[w2] - \text{mem} \cdot \text{Reg}[w0] \rightarrow \text{mem} \cdot \text{Reg}[w0] \]  
\[ 0x0804 - \text{mem}[0x0800] = 0x0800 - 0x1000 = 0xF804 \rightarrow \text{mem}[0x0800] \]  

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0800</td>
<td>0xF804</td>
</tr>
</tbody>
</table>

(7) dec2.b 0x0802  
Equation: \[ \text{mem}[0x0802] - 2 \rightarrow \text{mem}[0x0802] \]  
\[ 0x2000 - 2 = 0xFFFE \rightarrow \text{mem}[0x0802] \]  

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0802</td>
<td>0x30FE</td>
</tr>
</tbody>
</table>

(8) com.b W0, W1  
Equation: \[ \sim \text{Reg}[w0] \rightarrow \text{Reg}[w1] \]  
\[ \sim 0x0800 = 0xF7FF \rightarrow \text{Reg}[w1] \]  

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>0x80FE</td>
</tr>
</tbody>
</table>

(9) rrnc 0x0800  
Equation: \[ \text{mem}[0x0800] \gg \text{mem}[0x0800] \]  
\[ 0x1000 \gg 0x0800 \rightarrow \text{mem}[0x0800] \]  

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0800</td>
<td>0x0800</td>
</tr>
</tbody>
</table>

(10) btg 0x0802, #13  
Equation: \[ \text{mem}[0x0802] = 0010 0000 0000 0000 \rightarrow 0x0000 \rightarrow \text{mem}[0x0802] \]  

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0802</td>
<td>0x0000</td>
</tr>
</tbody>
</table>

**Question 2 (40 points):**

The following is the C program is given:

```c
uint16   n, p;
uint16   *a;
n = 0x1234;
a = &n;
p = *a;
```

Convert this program into Microchip assembly code with comments. The compiler should allocate a user space starting at 0x0800 in the data memory, generate a maximum of 5 assembly instructions in addition to the compiler directives and use registers W0 and W1 in SFR in the resulting assembly code. Show the contents of data memory and SFR (W0 and W1) after the executing the compiler directives in the resulting assembly code, after the equivalent assembly instruction to n = 0x1234, a = &n, and p = *a.
Question 2:

\( n = \) space 2
\( p = \) space 2

; \( w1 \) is used for a

\[
\begin{align*}
\text{mov} \ #0x1234, \ w0 & \quad \#0x1234 \rightarrow \text{mem}[n] \\
\text{mov} \ w0, \ n & \quad \text{which implements} \ n = 0x1234 \\
\text{mov} \ #n, \ w1 & \rightarrow \#n \ (\text{address of mem}[n]) \rightarrow \text{Reg}[w1] \text{ implements } a = \& n \\
\text{mov} \ [w1], \ w0 & \quad \text{mem}[\text{Reg}[w1]] = *a = \text{mem}[n] \rightarrow \text{mem}[p] \\
\text{mov} \ w0, \ p & \quad \text{implements } p = *a
\end{align*}
\]

After compiler directives:

\[
\begin{align*}
\text{Data Mem} & \\
0x0800 & - \\
0x0802 & - \\
& n \\
& p
\end{align*}
\]

After mov wo, n (\( n = 0x1234 \))

\[
\begin{align*}
\text{Data Mem} & \\
0x0800 & 0x1234 \\
0x0802 & 0x0800 \\
& n \\
& w0 0x1234
\end{align*}
\]

After mov \#n, w1 (\( a = \& n \))

\[
\begin{align*}
\text{Data Mem} & \\
0x0800 & 0x1234 \\
0x0802 & 0x0800 \\
& w0 \\
& w1 0x0800
\end{align*}
\]

After mov w0, p (\( p = *a \))

\[
\begin{align*}
\text{Data Mem} & \\
0x0800 & 0x1234 \\
0x0802 & 0x0800 \\
& w0 \\
& w1 0x0800
\end{align*}
\]
Question 3 (40 points):

The following Microchip assembly program is given (after the compiler directives):

```
0x0100    mov #0x0A00, W0
0x0102    call suba
0x0104    nop
0x0106    mov W4, 0x0800
0x0108    mov W3, 0x0802
0x010A    mov W2, 0x0804
0x010C    mov W1, 0x0806
    suba:
0x010E    call subb
0x0110    nop
0x0112    add W3, #0x0005, W4
0x0114    return
    subb:
0x0116    call subc
0x0118    nop
0x011A    sl W2, W2
0x011C    sub W2, #0x0002, W3
0x011E    return
    subc:
0x0120    mov W0, W1
0x0122    add W1, #0x0003, W2
0x0124    return
```

The stack initializes at 0x0900 in the data memory and the stack pointer address (SP) is stored at W15 in the SFR.

(1) Write the contents after nop instruction at address 0x0118 following call subc.

<table>
<thead>
<tr>
<th>Data Memory</th>
<th>SFR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0900</td>
<td>0x0A00</td>
</tr>
<tr>
<td>0x0902</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0904</td>
<td>0x0112</td>
</tr>
<tr>
<td>0x0906</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0908</td>
<td>0x011A</td>
</tr>
<tr>
<td>0x090A</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x090C</td>
<td></td>
</tr>
</tbody>
</table>

PC + 4 = 0x0116 + 4 = 0x011A \rightarrow \text{mem}[SP]

SP + 4 = 0x0908 + 4 = 0x090C \rightarrow \text{SP}

0x0120 \rightarrow PC
(2) Write the contents after return instruction at address 0x0114.

<table>
<thead>
<tr>
<th>Data Memory</th>
<th>SFR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0900</td>
<td>0x0106 ← SP</td>
</tr>
<tr>
<td>0x0902</td>
<td></td>
</tr>
<tr>
<td>0x0904</td>
<td></td>
</tr>
<tr>
<td>0x0906</td>
<td></td>
</tr>
<tr>
<td>0x0908</td>
<td></td>
</tr>
<tr>
<td>0x090A</td>
<td></td>
</tr>
<tr>
<td>0x090C</td>
<td></td>
</tr>
<tr>
<td>0x090E</td>
<td></td>
</tr>
</tbody>
</table>

W0 0x0A00
W1 0x0A00
W2 0x1406
W3 0x1404
W4 0x1409
W15 0x0900

SP - 4 = 0x0904 - 4 = 0x0900 → SP
mem[SP] = mem[0x0900] = 0x0106 → PC

Answer 3 workout space:
QUESTION 3:

0x0100: mov #0x0A00, w0
0x0102: call suba
0x0104: nop
0x0106: mov w4, 0x0800
0x0108: mov w3, 0x0802
0x010a: mov w2, 0x0804
0x010c: mov w1, 0x0806

suba:
0x010e: call subb
0x0110: nop
0x0112: add w3, #0x0005, w4
0x0114: return

subb:
0x0116: call subc
0x0118: nop
0x011a: sl w2, w2
0x011c: sub w2, #0x0002, w3
0x011e: return

subc:
0x0120: mov w0, w1
0x0122: add w1, #0x0003, w2
0x0124: return

0x0100 -> w0
0x010e -> w4
2 * (0xA003 - 1) + 5 -> w4
0x1404
2 * 0xA03 - 2 -> w3
0x1404
\[ y = f \{ z[k(x)] \} \quad \text{where} \quad k(x) = x + 3 \]
\[ z(k) = 2k - 2 \]
\[ f(z) = z + 5 \]

**call <label>**

- \( PC + 4 \rightarrow \text{mem}[SP] \) (return address after \text{nop})
- \( SP + 4 \rightarrow SP \)
- \( <\text{label}> \rightarrow PC \) (When \( SP = \text{reg}[W15] \))

**return**

- \( SP - 4 \rightarrow SP \)
- \( \text{mem}[SP] \rightarrow PC \)

Suppose stack initializes at data mem address 0x0900.

**After call suba & \text{nop}**

- \( PC + 4 = 0x0102 + 4 = 0x0106 \rightarrow \text{mem}[SP] \)
- \( SP + 4 = 0x0900 + 4 = 0x0904 \rightarrow SP \)

\[ W15 \quad 0x0900 \]
\[ W0 \quad 0x0A00 \]

\[ 0x010E \rightarrow PC \]
After call subb $n0

\[ PC + 4 = 0x010E + 4 = 0x0112 \rightarrow \text{mem}[SP] \]

\[ SP + 4 = 0x0904 + 4 = 0x0908 \rightarrow SP \]

\[ \langle \text{label} \rangle = 0x0116 \rightarrow PC \]

W15 \[ 0x0908 \]

W0 \[ 0x0A00 \]

---

After call subc $n0

\[ PC + 4 = 0x0116 + 4 = 0x011A \rightarrow \text{mem}[SP] \]

\[ SP + 4 = 0x0908 + 4 = 0x090c \rightarrow SP \]

\[ \langle \text{label} \rangle = 0x0120 \rightarrow PC \]

W15 \[ 0x090c \]

W0 \[ 0x0A00 \]
After add w1, #0x0003, w2
0x0a00 0x0b06
0x0a00 0x0000
0x0b1f 0x0d1c
0x0a00 0x0000
0x0a1a 0x0000
0x0a00 0x0000
0x0a00

SP

After the 1st return @ 0x0124
0x0a00 0x0b06 0x0000 0x0d1c
0x0a04 0x0112 0x0000
0x0a08 0x011a 0x0000

=w0 mem[sp]
=mem[0x0908]
=0x011a
=0x0908

SP-4

After sub w2, #0x0002, w3
0x0a00 0x0b06
0x0a00 0x0000
0x0b1f 0x0d1c
0x0a00 0x0000
0x0a1a 0x0000
0x0a00 0x0000
0x0b1c

After the 2nd return @ 0x011e
0x0a00 0x0b06 0x0000 0x0d1c
0x0a04 0x0112 0x0000
0x0a06 0x0600

=w0 mem[sp]
=mem[0x0904]
=0x0112
=0x0908

SP-4

After add w3, $0x0005, w4 @ 0x0112

After the 3rd return:

0x0900  0x0106  ← SP
0x0904  0x0112
0x0908  0x0000

0x0900  0x0106  ← SP
SP - 4 = 0x0904 - 4
= 0x0900 → SP
mem[SP] = mem[0x900]
= 0x0106 → PC

0x0900  0x0106  ← SP

0x0000  0x0A00
0x0A00  0x1406
0x1404
0x1409

0x0904  0x0900

After move w1, 0x0806:

0x0800  0x1409
0x0802  0x1404
0x0804  0x1406
0x0806  0x0A00

0x0900  0x0106  ← SP
**Question 1**  
**Part A (15 points)**

The following are the contents of the data memory and the register file (SFR).

<table>
<thead>
<tr>
<th>Data Memory</th>
<th>SFR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0800</td>
<td>W0</td>
</tr>
<tr>
<td>0x0802</td>
<td>W1</td>
</tr>
<tr>
<td>0x0804</td>
<td>W2</td>
</tr>
<tr>
<td>0x1000</td>
<td>0x0002</td>
</tr>
<tr>
<td>0x2000</td>
<td>0x0802</td>
</tr>
<tr>
<td>0x3000</td>
<td>0x0804</td>
</tr>
</tbody>
</table>

Write the operation for each instruction in terms of equation(s), the destination address and the contents of the result (in data memory or SFR):

- **mov.b [w1++], w2**  
  Equation:  
  \[
  \text{mem}_{\text{Reg}[w1]} = \text{mem}_{0x0802} = 0x2000 \rightarrow \text{Reg}[w1] + 2 \rightarrow 0x0802 + 2 = 0x0804 \rightarrow \text{Reg}[w1]
  \]
  
  **Address**
  
  **Contents**
  
  \[w2 \quad 0x0800\]

- **mov.b [w0+w1], w2**  
  Equation:  
  \[
  \text{mem}_{\text{Reg}[w0]+\text{Reg}[w1]} = \text{mem}_{0x0802} = 0x3000 \rightarrow \text{Reg}[w2]
  \]
  
  **Address**
  
  **Contents**
  
  \[w2 \quad 0x0800\]

- **mov.b #0x11, [w1]**  
  Equation:  
  Not possible!
  
  **Address**
  
  **Contents**
  
  ---

- **add.b w1, [w1], w2**  
  Equation:  
  \[
  \text{Reg}[w1] + \text{mem}_{\text{Reg}[w1]} = 0x0801 + \text{mem}_{0x0802} = 0x2802 \rightarrow \text{Reg}[w2]
  \]
  
  **Address**
  
  **Contents**
  
  \[w2 \quad 0x0802\]

- **sub.b [w1], [w2], w0**  
  Equation:  
  Not possible!
  
  **Address**
  
  **Contents**
  
  ---
Part B (15 points)

The following equation needs to be implemented using Microchip Assembly:

\[ \text{mem} [i] \to \text{mem} [j] \]

Use a maximum of 3 Assembly instructions to transfer the contents of memory at \(i\) to the memory location \(j\).

Question 2
Part A (15 points)

Implement moving \(0xA\) to the memory location at \(0x0800\) without using move instruction(s) in Microchip Assembly (use maximum 3 instructions).

Part B (15 points)

Construct the following C-program into a flowchart to be able to write Assembly code (don’t write the code).

```c
while ((i>j) && (k!=0) || (p<=q))
{
    f = a - b;
    g = 2a;
}
    f = a + b;
    g = 3a - 1;
```

DON’T WRITE IN THIS SPACE
**Question 1 - part B**

\[ \text{mem}[i] \rightarrow \text{mem}[j] \Rightarrow \]

**Option 1**

\[ \text{mov} \ #i, \ w1 \]

\[ \text{mov} \ i, \ w0 \]

\[ \text{mov} \ #j, \ w2 \]

\[ \text{mov} \ w0, \ j \]

\[ \text{mov} \ [w1], \ [w2] \]

**Option 2**

---

**Question 2 - part A**

\[ \text{setm.b} \ 0x0800 \]

\[ 1111 \ 1111 \]

\[ \text{bclr.b} \ 0x0800, \ #4 \]

\[ 1110 \ 0111 \]

\[ \text{bclr.b} \ 0x0800, \ #6 \]

\[ 0100 \ 1111 = 0xA5 \text{ - mem}(0x0800) \]

---

**Question 2 - part B**

\[ \text{while} \ ((i > j) \ \&\&(k != 0)) \!\! \\
\]

\[ f = a - b \]

\[ g = 2a \]

\[ f = a + b \]

\[ g = 3a - 1 \]

---

**Diagram**

The diagram shows a flowchart with decision points and loops. The decision points are labeled with conditions and paths labeled with actions and operations.
Question 3 (40 points)

The following C program implements a for-loop working on an array.

```c
unsigned short i;
unsigned short j[4] = {0x12, 0x34, 0x11, 0x33};
unsigned short k[4];
for (i = 0; i < 4; i ++)
k[i] = i + j[i];
```

Use the flowchart below to indicate the compiler directives, assignments, equation(s) and decisions in each box that correspond to the instructions of this program. Write Microchip Assembly code next to each box. The dashed line boundary separates each code segment from each other. Do not place the flowchart and the code in separate places. Use w1 for i, w2 for j and w3 for k.
Compiler directives

Assignments for RF

i → Reg[w1]
j → Reg[w2]
k → Reg[w3]

label top

Decision box

Reg[w1] ≥ 4

else

label end

Equations

i + mem[j] → mem[k]

Assignments for memory

i + 1 → i
j + 1 → j
k + 1 → k

clr.b w1
mov.b #j, w2
mov.b #k, w3

cp.b w1, #4
bra GEU, end

add w1, [w2++], [w3++]

inc.b w1, w1
bra top
end =

j  . space 4 x 1
k  . space 4 x 1
i  . space 1
**Question 1 (30 points)**

A 32-bit RISC CPU needs to execute the following flow chart:

- Fetch A from memory address 100
- Fetch B from memory address 101

\[
X = \frac{A + B}{3}
\]

- Store X to memory address 200

The ALU does not contain a multiplier or a divider unit in this CPU. The register file has only 3 registers with Reg [R0] = 0 as shown below.

<table>
<thead>
<tr>
<th>R0</th>
<th>R1</th>
<th>R2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Using the basic RISC instruction set (not the PIC 24 instructions please), write an assembly code that implements this flow chart. Indicate the computation error caused by your program.

\[
\text{Error} = 0.333 - 0.328 = 0.005 \text{ or } 1.5\%
\]

<table>
<thead>
<tr>
<th>RISC Instruction</th>
<th>Equation</th>
<th>Reg[R1]</th>
<th>Reg[R2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOAD R0, R1, 100</td>
<td>mem[100] = A \rightarrow \text{Reg[R1]}</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>LOAD R0, R2, 101</td>
<td>mem[101] = B \rightarrow \text{Reg[R2]}</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>ADD R1, R2, R1</td>
<td>A + B \rightarrow \text{Reg[R1]}</td>
<td>A + B</td>
<td>B</td>
</tr>
<tr>
<td>SRI R1, R1, 2</td>
<td>(A+B)/4 \rightarrow \text{Reg[R1]}</td>
<td>(A+B)/4</td>
<td>B</td>
</tr>
<tr>
<td>SRI R1, R2, 2</td>
<td>(A+B)/16 \rightarrow \text{Reg[R2]}</td>
<td>(A+B)/4</td>
<td>(A+B)/16</td>
</tr>
<tr>
<td>ADD R1, R2, R1</td>
<td>(A+B)/4 + (A+B)/16 \rightarrow \text{Reg[R1]}</td>
<td>(A+B)/4 + (A+B)/16</td>
<td>(A+B)/16</td>
</tr>
<tr>
<td>SRI R2, R2, 2</td>
<td>(A+B)/64 \rightarrow \text{Reg[R2]}</td>
<td>(A+B)/4 + (A+B)/16</td>
<td>(A+B)/64</td>
</tr>
<tr>
<td>ADD R1, R2, R2</td>
<td>X \rightarrow \text{Reg[R2]}</td>
<td>A + X</td>
<td>X</td>
</tr>
<tr>
<td>STORE R2, R0, 200</td>
<td>Reg[R2] = X \rightarrow \text{mem[200]}</td>
<td>A + B + A + B/16</td>
<td>X</td>
</tr>
</tbody>
</table>
Question 2 (30 points)

(a) Using the PIC-24 instruction set, construct an Assembly program that swaps the bytes, 0xAB and 0xCD, in register w1 and writes the result into the register w2 as shown below. You are allowed to use only these 2 registers and not to exceed 3 instructions in the entire program.

<table>
<thead>
<tr>
<th>PIC24 Instruction</th>
<th>Dest. Reg</th>
<th>MSByte</th>
<th>LSByte</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1 w1, #8, w2</td>
<td>w2</td>
<td>0xCD</td>
<td>0x00</td>
</tr>
<tr>
<td>lsr w1, #8, w1</td>
<td>w1</td>
<td>0x00</td>
<td>0xAB</td>
</tr>
<tr>
<td>mov. b w1, w2</td>
<td>w2</td>
<td>0xCD</td>
<td>0xAB</td>
</tr>
</tbody>
</table>

(b) Using the PIC-24 instruction set, construct an Assembly program that swaps the bytes, 0xAB and 0xCD, in register w1 and writes the result into the memory location 0x0800 as shown below. You are allowed to use only the register w1 and mem{0x0800} and not to exceed 3 instructions in the entire program.

<table>
<thead>
<tr>
<th>PIC24 Instruction</th>
<th>Dest. Reg/Mem Addr</th>
<th>MSByte</th>
<th>LSByte</th>
</tr>
</thead>
<tbody>
<tr>
<td>mov. b w1, 0x0800</td>
<td>0x0800</td>
<td>0xCD</td>
<td>0x00</td>
</tr>
<tr>
<td>lsr w1, #8, w1</td>
<td>w1</td>
<td>0x00</td>
<td>0xAB</td>
</tr>
<tr>
<td>mov. b w1, 0x0800</td>
<td>0x0800</td>
<td>0xCD</td>
<td>0xAB</td>
</tr>
</tbody>
</table>
Question 3 (40 points)

Construct the architectural diagram of the 16-bit, 4-stage PIC-24 CPU that executes only the following instructions:

\[
\begin{align*}
\text{mov w1} & , [w2] \\
\text{mov} & \quad \text{w1} \quad [w2] \\
\text{mov}[w1], w2 & \\
\text{mov} & \quad [w1] \quad w2
\end{align*}
\]

Include the cumulative hardware that executes these 2 instructions in a 4-stage data-path. Indicate all memory port names, label all the data and address names, bit widths, mux selection inputs to direct the data-flow (if any) clearly. Make sure to rotate the page in scenic view and use ruler. Neatness and clarity of your architectural will gain you points!

DON'T WRITE ANYTHING HERE!
USE THE NEXT PAGE
$\text{mov} [\text{m}], w2 \in \text{mov} [\text{m}']$  
\[ \text{mov} [\text{m}'] \in [\text{m}'] \wedge \text{mov} [\text{m}'] \in [\text{m}'] \]