

Homework 4

Brandon Hosley
Mike Davis

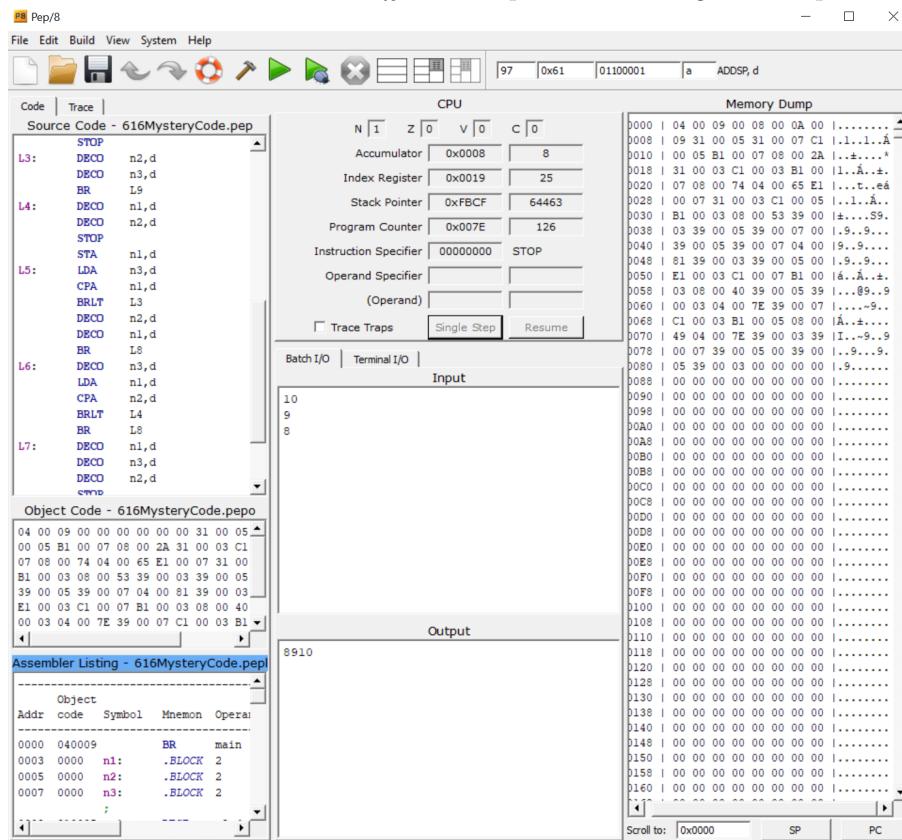
Homework 4

Problem 1

Run the mystery program from Fig 6.16 with the values supplied in the help solution of Pep8 and some of your own.

1.a

Show 3 screen shots with different inputs, including the output using Batch I/O.



Pep/8

File Edit Build View System Help

CPU

N	I	Z	V	C
0000	00	00	00	00
Accumulator	0x004B		75	
Index Register	0x0019		25	
Stack Pointer	0xFBCF		64463	
Program Counter	0x004F		79	
Instruction Specifier	00000001		RETTR	
Operand Specifier				
(Operand)				

Trace Traps Single Step Resume

Memory Dump

```

0000 | 04 00 09 00 4B 00 64 00 |....K.d.▲
0008 | 32 31 00 05 31 00 07 C1 |21..1..Å
0010 | 00 05 B1 00 07 08 00 2A |..±..±..*
0018 | 31 00 03 C1 00 03 B1 00 |1..Å..±.
0020 | 07 08 00 74 04 00 65 E1 |...t..éä
0028 | 00 07 31 00 03 C1 00 05 |..1..Å..
0030 | B1 00 03 08 00 53 39 00 |z...S9.
0038 | 03 39 00 05 39 00 07 00 |1..-9..9...
0040 | 39 00 05 39 00 07 04 00 |1..9..9...
0048 | 81 39 00 03 39 00 05 00 |1..9..9...
0050 | E1 00 03 C1 00 07 B1 00 |1..Å..±.
0058 | 03 08 00 40 39 00 05 39 |...89..9
0060 | 00 03 04 00 7E 39 00 07 |.....-9..
0068 | C1 00 03 B1 00 05 08 00 |Å..±....
0070 | 49 04 00 7E 39 00 03 39 |I..-9..9
0078 | 00 07 39 00 05 00 39 00 |1..-9..9...
0080 | 05 39 00 03 00 00 00 00 |1..9.....
0088 | 00 00 00 00 00 00 00 00 |1.....
0090 | 00 00 00 00 00 00 00 00 |1.....
0098 | 00 00 00 00 00 00 00 00 |1.....
00A0 | 00 00 00 00 00 00 00 00 |1.....
00A8 | 00 00 00 00 00 00 00 00 |1.....
00B0 | 00 00 00 00 00 00 00 00 |1.....
00B8 | 00 00 00 00 00 00 00 00 |1.....
00C0 | 00 00 00 00 00 00 00 00 |1.....
00C8 | 00 00 00 00 00 00 00 00 |1.....
00D0 | 00 00 00 00 00 00 00 00 |1.....
00D8 | 00 00 00 00 00 00 00 00 |1.....
00E0 | 00 00 00 00 00 00 00 00 |1.....
00E8 | 00 00 00 00 00 00 00 00 |1.....
00F0 | 00 00 00 00 00 00 00 00 |1.....
00F8 | 00 00 00 00 00 00 00 00 |1.....
0100 | 00 00 00 00 00 00 00 00 |1.....
0108 | 00 00 00 00 00 00 00 00 |1.....
0110 | 00 00 00 00 00 00 00 00 |1.....
0118 | 00 00 00 00 00 00 00 00 |1.....
0120 | 00 00 00 00 00 00 00 00 |1.....
0128 | 00 00 00 00 00 00 00 00 |1.....
0130 | 00 00 00 00 00 00 00 00 |1.....
0138 | 00 00 00 00 00 00 00 00 |1.....
0140 | 00 00 00 00 00 00 00 00 |1.....
0148 | 00 00 00 00 00 00 00 00 |1.....
0150 | 00 00 00 00 00 00 00 00 |1.....
0158 | 00 00 00 00 00 00 00 00 |1.....
0160 | 00 00 00 00 00 00 00 00 |1.....

```

Batch I/O Terminal I/O Input

100
50
75

Output

```
5075100
```

Scroll to: 0x0000 SP PC

Pep/8

File Edit Build View System Help

CPU

N	I	Z	V	C
0000	00	00	00	00
Accumulator	0x000A		10	
Index Register	0x0019		25	
Stack Pointer	0xFBCF		64463	
Program Counter	0x003F		63	
Instruction Specifier	00000001		RETTR	
Operand Specifier				
(Operand)				

Trace Traps Single Step Resume

Memory Dump

```

0000 | 04 00 09 00 0A 00 0A 01 |.....
0008 | F4 31 00 05 31 00 07 C1 |61..1..Å
0010 | 00 05 B1 00 07 08 00 2A |..±..±..*
0018 | 31 00 03 C1 00 03 B1 00 |1..Å..±.
0020 | 07 08 00 74 04 00 65 E1 |...t..éä
0028 | 00 07 31 00 03 C1 00 05 |..1..Å..
0030 | B1 00 03 08 00 53 39 00 |z...S9.
0038 | 03 39 00 05 39 00 07 00 |1..-9..9...
0040 | 39 00 05 39 00 07 04 00 |1..9..9...
0048 | 81 39 00 03 39 00 05 00 |1..9..9...
0050 | E1 00 03 C1 00 07 B1 00 |1..Å..±.
0058 | 03 08 00 40 39 00 05 39 |...89..9
0060 | 00 03 04 00 7E 39 00 07 |.....-9..
0068 | C1 00 03 B1 00 05 08 00 |Å..±....
0070 | 49 04 00 7E 39 00 03 39 |I..-9..9
0078 | 00 07 39 00 05 00 39 00 |1..-9..9...
0080 | 05 39 00 03 00 00 00 00 |1..9.....
0088 | 00 00 00 00 00 00 00 00 |1.....
0090 | 00 00 00 00 00 00 00 00 |1.....
0098 | 00 00 00 00 00 00 00 00 |1.....
00A0 | 00 00 00 00 00 00 00 00 |1.....
00A8 | 00 00 00 00 00 00 00 00 |1.....
00B0 | 00 00 00 00 00 00 00 00 |1.....
00B8 | 00 00 00 00 00 00 00 00 |1.....
00C0 | 00 00 00 00 00 00 00 00 |1.....
00C8 | 00 00 00 00 00 00 00 00 |1.....
00D0 | 00 00 00 00 00 00 00 00 |1.....
00D8 | 00 00 00 00 00 00 00 00 |1.....
00E0 | 00 00 00 00 00 00 00 00 |1.....
00E8 | 00 00 00 00 00 00 00 00 |1.....
00F0 | 00 00 00 00 00 00 00 00 |1.....
00F8 | 00 00 00 00 00 00 00 00 |1.....
0100 | 00 00 00 00 00 00 00 00 |1.....
0108 | 00 00 00 00 00 00 00 00 |1.....
0110 | 00 00 00 00 00 00 00 00 |1.....
0118 | 00 00 00 00 00 00 00 00 |1.....
0120 | 00 00 00 00 00 00 00 00 |1.....
0128 | 00 00 00 00 00 00 00 00 |1.....
0130 | 00 00 00 00 00 00 00 00 |1.....
0138 | 00 00 00 00 00 00 00 00 |1.....
0140 | 00 00 00 00 00 00 00 00 |1.....
0148 | 00 00 00 00 00 00 00 00 |1.....
0150 | 00 00 00 00 00 00 00 00 |1.....
0158 | 00 00 00 00 00 00 00 00 |1.....
0160 | 00 00 00 00 00 00 00 00 |1.....

```

Batch I/O Terminal I/O Input

10
500
10

Output

```
1010500
```

Scroll to: 0x0000 SP PC

1.b

State in one sentence, what does this program do?

This program takes a series of numbers and returns them in numeric order.

1.c

*Modify the program to make the output clearer – Describe – Do not paste the code.
Show the same 3 screen shots with the modification.*

In order to make the output more clear I have decided to make better use of spacing on the output side. This was simply a matter of adding a reference to the \n character and calling it after each output.

```

Pep8
File Edit Build View System Help
Code Trace | Listing Trace
CPU
N [1] Z [0] V [0] C [0]
Accumulator [0x0008] 8
Index Register [0x0019] 25
Stack Pointer [0xFBCF] 64463
Program Counter [0x00A3] 163
Instruction Specifier [00000001] RETTR
Operand Specifier [ ]
(Operand) [ ]
Trace Traps Single Step Resume
Batch I/O Terminal I/O | Input
10
9
8
Output
8
9
10
Memory Dump
0000 | 04 00 0B 00 08 00 0A 00 |.....|
0008 | 09 0A 00 31 00 05 31 00 |..,1.,1.|
0010 | 07 C1 00 05 B1 00 07 08 |.Ä.,z..|
0018 | 00 2C 31 00 03 C1 00 03 |.,1.,Ä..|
0020 | B1 00 07 08 00 94 04 00 |.....|
0028 | 82 E1 00 07 31 00 03 C1 |.Ä.,1.,J|
0030 | 00 05 B1 00 03 08 00 6A |..,z..,|
0038 | 39 00 03 41 00 09 39 00 |9.,Ä.,9.|
0040 | 05 41 00 09 39 00 07 41 |.Ä.,9.,J|
0048 | 00 09 00 39 00 05 41 00 |..,9.,Ä.|
0050 | 09 39 00 07 51 00 09 04 |1.,9.,Q..|
0058 | 00 AD 39 00 03 51 00 09 |1.,9.,Q..|
0060 | 39 00 05 51 00 09 00 E1 |1.,9.,Q..,|
0068 | 00 03 C1 00 07 B1 00 03 |..,Ä.,z..|
0070 | 00 00 48 39 00 05 51 00 |1.,K9.,Q..|
0078 | 09 39 00 03 51 00 09 04 |1.,9.,Q..|
0080 | 00 A7 39 00 07 51 00 09 |1.,$,Q..|
0088 | C1 00 03 B1 00 05 08 00 |Ä.,z..|
0090 | 5A 04 00 A7 39 00 03 51 |Z.,$,Q..,|
0098 | 00 09 39 00 07 51 00 09 |1.,9.,Q..|
00A0 | 39 00 05 51 00 09 00 E1 |1.,9.,Q..,|
00A8 | 00 05 51 00 09 39 00 03 |1.,Q..,9..|
00B0 | 51 00 09 00 00 00 00 00 |Q,.....|
00B8 | 00 00 00 00 00 00 00 00 |.....|
00C0 | 00 00 00 00 00 00 00 00 |.....|
00C8 | 00 00 00 00 00 00 00 00 |.....|
00D0 | 00 00 00 00 00 00 00 00 |.....|
00D8 | 00 00 00 00 00 00 00 00 |.....|
00E0 | 00 00 00 00 00 00 00 00 |.....|
00E8 | 00 00 00 00 00 00 00 00 |.....|
00F0 | 00 00 00 00 00 00 00 00 |.....|
00F8 | 00 00 00 00 00 00 00 00 |.....|
0100 | 00 00 00 00 00 00 00 00 |.....|
0108 | 00 00 00 00 00 00 00 00 |.....|
0110 | 00 00 00 00 00 00 00 00 |.....|
0118 | 00 00 00 00 00 00 00 00 |.....|
0120 | 00 00 00 00 00 00 00 00 |.....|
0128 | 00 00 00 00 00 00 00 00 |.....|
0130 | 00 00 00 00 00 00 00 00 |.....|
0138 | 00 00 00 00 00 00 00 00 |.....|
0140 | 00 00 00 00 00 00 00 00 |.....|
0148 | 00 00 00 00 00 00 00 00 |.....|
0150 | 00 00 00 00 00 00 00 00 |.....|
0158 | 00 00 00 00 00 00 00 00 |.....|
0160 | 00 00 00 00 00 00 00 00 |.....|

```

Pep/8

File Edit Build View System Help

CPU Registers:

N	Z	V	C
0	1	0	0

Accumulator: 0x004B, Index Register: 0x0019, Stack Pointer: 0xFBCF, Program Counter: 0x0067, Instruction Specifier: 00000000 STOP, Operand Specifier: (Operand)

Trace Traps: Single Step, Resume

Memory Dump:

```

0000 | 04 00 0B 00 4B 00 64 00 1...K.d
0008 | 32 0A 00 31 00 05 31 00 12..1..1.
0010 | 07 C1 00 05 B1 00 07 08 1.A..z...
0018 | 00 2C 31 00 03 C1 00 03 1..1..A..
0020 | B1 00 07 08 09 94 04 00 1.....
0028 | 82 E1 00 07 31 00 03 C1 1..A..1..J
0030 | 00 05 B1 00 03 08 00 6A 1..z...1
0038 | 39 00 03 41 00 09 39 00 9..A..9.
0040 | 05 41 00 09 39 00 07 41 1..A..9..J
0048 | 00 09 00 39 00 05 41 00 1..z..A...
0050 | 09 39 00 07 51 00 09 04 1..9..Q...
0058 | 00 AD 39 00 03 51 00 09 1..9..Q...
0060 | 39 00 05 51 00 09 00 E1 1..9..Q..4
0068 | 00 03 C1 00 07 B1 00 03 1..A..z...
0070 | 08 00 4B 39 00 05 51 00 1..K9..Q.
0078 | 09 39 00 03 51 00 09 04 1..9..Q...
0080 | 00 27 39 00 07 51 00 09 1..$9..Q...
0088 | C1 00 03 B1 00 05 08 00 1..z...1...
0090 | SA 04 00 A7 39 00 03 51 1..Z..$9..1<
0098 | 00 09 39 00 07 51 00 09 1..9..Q...
00A0 | 39 00 05 51 00 09 00 39 1..9..Q..5
00A8 | 00 05 51 00 09 39 00 03 1..Q..9...
00B0 | 51 00 00 00 00 00 00 00 1.....
00B8 | 00 00 00 00 00 00 00 00 1.....
00C0 | 00 00 00 00 00 00 00 00 1.....
00C8 | 00 00 00 00 00 00 00 00 1.....
00D0 | 00 00 00 00 00 00 00 00 1.....
00D8 | 00 00 00 00 00 00 00 00 1.....
00E0 | 00 00 00 00 00 00 00 00 1.....
00E8 | 00 00 00 00 00 00 00 00 1.....
00F0 | 00 00 00 00 00 00 00 00 1.....
00F8 | 00 00 00 00 00 00 00 00 1.....
0100 | 00 00 00 00 00 00 00 00 1.....
0108 | 00 00 00 00 00 00 00 00 1.....
0110 | 00 00 00 00 00 00 00 00 1.....
0118 | 00 00 00 00 00 00 00 00 1.....
0120 | 00 00 00 00 00 00 00 00 1.....
0128 | 00 00 00 00 00 00 00 00 1.....
0130 | 00 00 00 00 00 00 00 00 1.....
0138 | 00 00 00 00 00 00 00 00 1.....
0140 | 00 00 00 00 00 00 00 00 1.....
0148 | 00 00 00 00 00 00 00 00 1.....
0150 | 00 00 00 00 00 00 00 00 1.....
0158 | 00 00 00 00 00 00 00 00 1.....
0160 | 00 00 00 00 00 00 00 00 1.....

```

Batch I/O | Terminal I/O | Input:

```

100
50
75

```

Output:

```

50
75
100

```

Scroll to: 0x0000 SP PC

Pep/8

File Edit Build View System Help

CPU Registers:

N	Z	V	C
0	1	0	1

Accumulator: 0x000A, Index Register: 0x0019, Stack Pointer: 0xFBCF, Program Counter: 0x004A, Instruction Specifier: 00000001 RETTR, Operand Specifier: (Operand)

Trace Traps: Single Step, Resume

Memory Dump:

```

0000 | 04 00 0B 00 0A 00 0A 01 1.....
0008 | F4 0A 00 31 00 05 31 00 16..1..1.
0010 | 07 C1 00 05 B1 00 07 08 1..A..z...
0018 | 00 2C 31 00 03 C1 00 03 1..1..A..
0020 | B1 00 07 08 09 94 04 00 1.....
0028 | 82 E1 00 07 31 00 03 C1 1..A..1..J
0030 | 00 05 B1 00 03 08 00 6A 1..z...1
0038 | 39 00 03 41 00 09 39 00 9..A..9.
0040 | 05 41 00 09 39 00 07 41 1..A..9..J
0048 | 00 09 00 39 00 05 41 00 1..z..A...
0050 | 09 39 00 07 51 00 09 04 1..9..Q...
0058 | 00 AD 39 00 03 51 00 09 1..K9..Q...
0060 | 39 00 05 51 00 09 00 E1 1..9..Q..4
0068 | 00 03 C1 00 07 B1 00 03 1..A..z...
0070 | 08 00 4B 39 00 05 51 00 1..K9..Q.
0078 | 09 39 00 03 51 00 09 04 1..9..Q...
0080 | 00 27 39 00 07 51 00 09 1..$9..Q...
0088 | C1 00 03 B1 00 05 08 00 1..z...1...
0090 | SA 04 00 A7 39 00 03 51 1..Z..$9..1<
0098 | 00 09 39 00 07 51 00 09 1..9..Q...
00A0 | 39 00 05 51 00 09 00 39 1..9..Q..5
00A8 | 00 05 51 00 09 39 00 03 1..Q..9...
00B0 | 51 00 00 00 00 00 00 00 1.....
00B8 | 00 00 00 00 00 00 00 00 1.....
00C0 | 00 00 00 00 00 00 00 00 1.....
00C8 | 00 00 00 00 00 00 00 00 1.....
00D0 | 00 00 00 00 00 00 00 00 1.....
00D8 | 00 00 00 00 00 00 00 00 1.....
00E0 | 00 00 00 00 00 00 00 00 1.....
00E8 | 00 00 00 00 00 00 00 00 1.....
00F0 | 00 00 00 00 00 00 00 00 1.....
00F8 | 00 00 00 00 00 00 00 00 1.....
0100 | 00 00 00 00 00 00 00 00 1.....
0108 | 00 00 00 00 00 00 00 00 1.....
0110 | 00 00 00 00 00 00 00 00 1.....
0118 | 00 00 00 00 00 00 00 00 1.....
0120 | 00 00 00 00 00 00 00 00 1.....
0128 | 00 00 00 00 00 00 00 00 1.....
0130 | 00 00 00 00 00 00 00 00 1.....
0138 | 00 00 00 00 00 00 00 00 1.....
0140 | 00 00 00 00 00 00 00 00 1.....
0148 | 00 00 00 00 00 00 00 00 1.....
0150 | 00 00 00 00 00 00 00 00 1.....
0158 | 00 00 00 00 00 00 00 00 1.....
0160 | 00 00 00 00 00 00 00 00 1.....

```

Batch I/O | Terminal I/O | Input:

```

10
500
10

```

Output:

```

10
10
500

```

Scroll to: 0x0000 SP PC

1.d

Is this spaghetti code or structured code? If it was the other type, would it be easier or harder to modify?

This is a bit of spaghetti code. If it had been written in a more structured manner it would have been much easier to modify. The modification in the previous part of the problem was doable by simply following each print command, which made the modification easier, but to restructure the code would have been a much more difficult task. However, once the code had been restructured further modification would have most likely been much easier.

Problem 2

Translate the program below into PEP/8 assembly language

- Start with Assembly code for Fig 6.36 (Use Pep8 help)
- Change to output array in same order as input
- Add function twoVect
- Pass array as parameter as shown in Fig 6.38
- Use trace tags on all variables.

2.a

Comment lines of the source code to trace the C++ code. Cut & paste the Source Code Listing into your assignment document.

```

BR          main
;
;***** twoVect
v:    .EQUATE 6      ;local variable #2d4a
n:    .EQUATE 2      ;local variable #2d
k:    .EQUATE 0      ;local variable #2d
twoVect: SUBSP 2,i   ;allocate k
          LDX 0,i    ;for(k = 0,
          STX k,s    ;|
for0:   CPX 4,i    ; k < n,
          BRGE endFor0 ; |
;
ASLX          ; an integer is two bytes
LDA v,sxf    ; k[v]
ASLA          ; * 2
STA v,sxf    ; => k[v]
;
LDX k,s      ; k++)
ADDX 1,i      ; |
STX k,s      ; |
BR for0      ; End For0
endFor0: RET2    ;
;
```

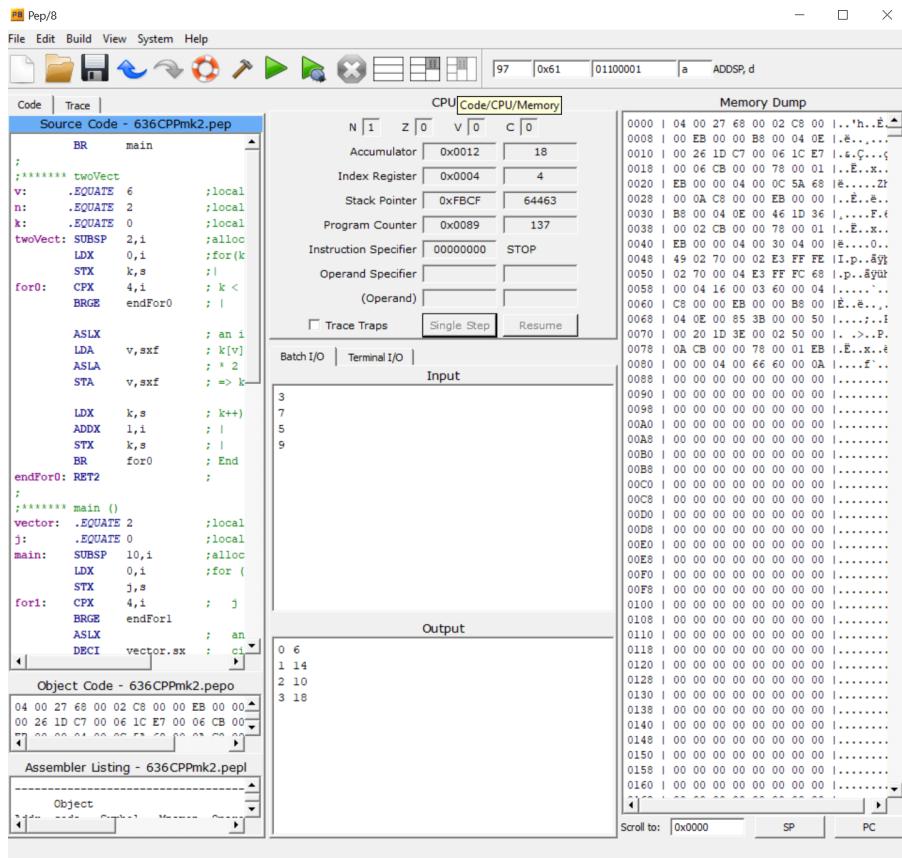
```

;***** main ()
vector: .EQUATE 2      ;local variable #2d4a
j:     .EQUATE 0      ;local variable #2d
main:   SUBSP 10,i    ;allocate #vector #j
        LDX 0,i       ;for (j = 0
        STX j,s
for1:   CPX 4,i       ; j < 4
        BRGE endFor1
        ASLX           ; an integer is two bytes
        DECI vector,sx ; cin » vector[j]
        LDX j,s       ; j++)
        ADDX 1,i
        STX j,s
        BR for1
endFor1: BR tvCall
;***** Call: twoVect
;
tvCall: MOVSPA         ;push address of vector
        ADDA vector,i
        STA -2,s
        MOVSPA         ;push address of n
        ADDA 4,i
        STA -4,s
        SUBSP 4,i      ;push n
        CALL twoVect,i ;Call twoVect
        ADDSP 4,i      ;
;
;; Print Loop
        LDX 0,i       ;for (j = 0
        STX j,s
for2:   CPX 4,i       ; j > 4
        BRGE endFor2
        DECO j,s       ; cout « j
        CHARO ',',i   ; « ,
        ASLX           ; an integer is two bytes
        DECO vector,sx ; « vector[j]
        CHARO '/n',i   ; « endl
        LDX j,s       ; j-
        ADDX 1,i
        STX j,s
        BR for2
endFor2: ADDSP 10,i    ;deallocate #j #vector
        STOP
.END

```

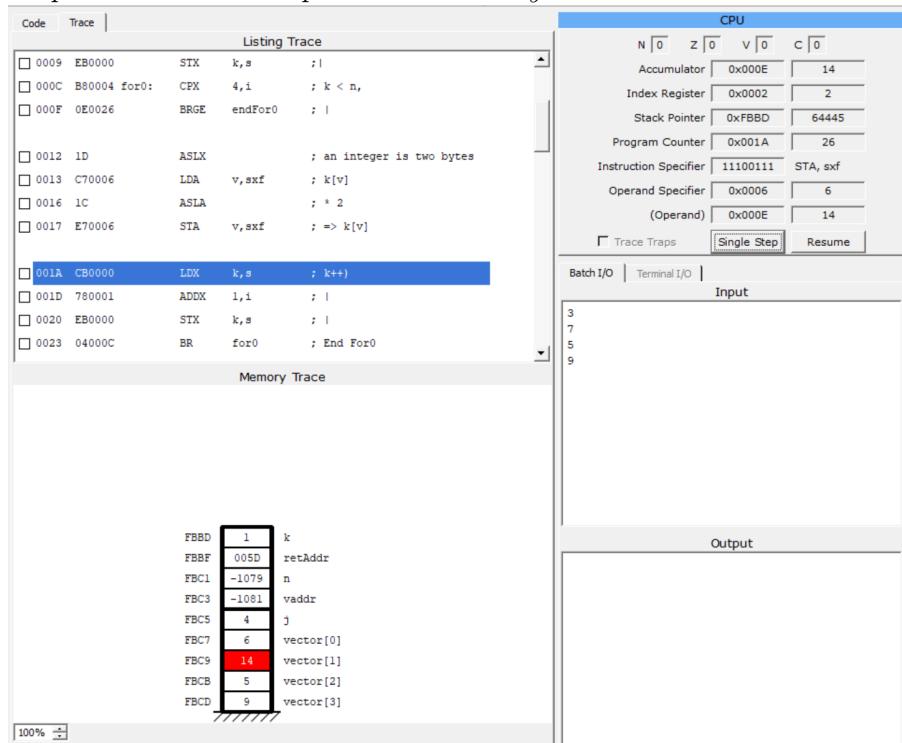
2.b

Run for a set of 4 inputs and paste a screen shot of the Output area of PEP/8.



2.c

Step thru & Cut and paste the memory trace when in the twoVect function.



Problem 3

Translate the program below into PEP/8 assembly language.

- Use a jump table to implement the switch statement.
- Use trace tags on all variables.
- For invalid scores, output should be the same as the C++ program.
- Add something to the output that makes this program uniquely yours.
- The variable *finish* needs to be local.
- This is similar to Fig 6.40

3.a

Comment lines of the source code to trace the C++ code. Cut & paste the Source Code Listing into your assignment document.

```

BR      main
;
;***** main ()
finish: .EQUATE 0      ;local variable #2d
main:   SUBSP   2,i    ;allocate #finish
        STRO    msgIn,d ;cout « "Enter your score: 1, 2, 3, 4, or 5"
        CHARO   '/n',i  ;cout « endl
        DECI    finish,s ;cin » Guess
        LDX     finish,s ;switch (Guess)
        SUBX   1,i      ;subtract 1
        ASLX
        BR      finishJT,x ;addresses occupy two bytes
finishJT: .ADDRSS case0
          .ADDRSS case1
          .ADDRSS case2
          .ADDRSS case3
          .ADDRSS case4
case0:   STRO    msg0,d ;cout « "You're the first!"
          BR      endCase ;break
case1:   STRO    msg1,d ;cout « "You're the first loser!"
          BR      endCase ;break
case2:   STRO    msg2,d ;cout « "Try Harder!"
          BR      endCase ;break
case3:   STRO    msg2,d ;cout « "Try Harder!"
          BR      endCase ;break
case4:   STRO    msg3,d ;cout « "You weren't even Competing!"
endCase: CHARO   '/n',i  ;cout « endl
          ADDSP   2,i      ;deallocate #finish
          STOP
msgIn:  .ASCII   "Enter your score: 1, 2, 3, 4, or 5 /x00"
msg0:   .ASCII   "You're the first! /x00"
msg1:   .ASCII   "You're the first loser! /x00"
msg2:   .ASCII   "Try Harder! /x00"
msg3:   .ASCII   "You weren't even Competing! /x00"
.END

```

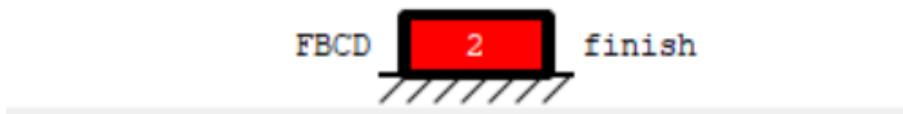
3.b

Run for each score and paste a screen shot of each of the PEP/8 Output area.

<p>Batch I/O Terminal I/O Input/Output</p> <pre>Enter your score: 1, 2, 3, 4, or 5 1 You're the first!</pre> <p>Input/Output</p> <pre>Enter your score: 1, 2, 3, 4, or 5 3 Try Harder!</pre>	<p>Batch I/O Terminal I/O Input/Output</p> <pre>Enter your score: 1, 2, 3, 4, or 5 2 You're the first loser!</pre> <p>Input/Output</p> <pre>Enter your score: 1, 2, 3, 4, or 5 4 Try Harder!</pre>
<p>Batch I/O Terminal I/O Input/Output</p> <pre>Enter your score: 1, 2, 3, 4, or 5 5 You weren't even Competing!</pre>	

3.c

Step thru & Cut and paste the memory trace at any point.



Problem 4

Write a C++ program that inputs a lower case character, converts it to an upper case character using the function below and increments it to the next character (i.e. B will be changed to C). If Z is entered it should be changed to A. If a non-letter character is entered, it should not be changed.

- A character that is not a letter should be returned as is.
- Character variables will need character trace tags.
- Hint: characters only use one byte of storage and should be manipulated with byte instructions.
- Add something to the output that makes this program uniquely yours.
- Then translate it to Assembly language.

4.a

Cut and paste your C++ Source Code into your assignment document.

```

/*
CSC - Homework 4 - Problem 4
Author: Brandon Hosley
Date: 2018 10 13
*/

#include "pch.h"
#include <iostream>
using namespace std;

char uppercase(char ch) {
    if ((ch >= 'a') && (ch <= 'z'))
    {
        return ch - 'a' + 'A';
    }
    else
    {
        return ch;
    }
}

char increment(char ch)
{
    if (ch == 'z' || ch == 'Z')
    {
        return 'A';
    }
    else
    {
        return ++ch;
    }
}

```

```
int main()
{
    char ch;
    cout << "Please input character:" << endl;
    cin >> ch;
    ch = uppercase(ch);
    ch = increment(ch);
    cout << ch;
    return 0;
}
```

4.b

Comment lines of the source code to trace the C++ code. Cut & paste the Assembly Source Code Listing into your assignment document.

```

        br          main
;

;***** uppercase(char ch)
upcase:    LDBYTEA  ch,d      ;if (
            CPA       'a',i      ;ch >= 'a'
            BRLT      endUpper
            LDBYTEA  ch,d
            CPA       'z',i      ;ch <= 'z'
            BRGT      endUpper  ;
            LDBYTEA  ch,d
            SUBA      32,i      ; ch = ch - 'a' + 'A'
            STBYTEA  ch,d
endUpper:  RET0

;
;***** increment(char ch)
incr:      LDBYTEA  ch,d      ;if(
            CPA       'Z',i      ; ch == 'Z'
            BREQ      else
            LDBYTEA  ch,d
            CPA       'z',i      ; ch == 'z'
            BREQ      else
            ADDA      1,i       ;++ch
            STBYTEA  ch,d
            br       endIncr
else:      LDBYTEA  'A',i      ; ch = 'A'
            STBYTEA  ch,d
            endIncr:  RET0

;
;***** main()
ch:        .EQUATE  0          ;local variable #1c
main:      SUBSP    1,i       ;allocate #ch
            STRO     msg,d     ;cout « "Please input your character:"
            CHARO    '/n',i     ;cout « endl
            CHARI    ch,d      ;cin » ch
            CALL     upcase,i   ;upcase(ch)
            CALL     incr,i     ;increment(ch)
            CHARO    ch,d      ;cout « ch
;

ADDSP    1,i       ;deallocate #ch
STOP

;***** Constants
msg:      .ASCII    "Please input your character: /x00"
.END

```

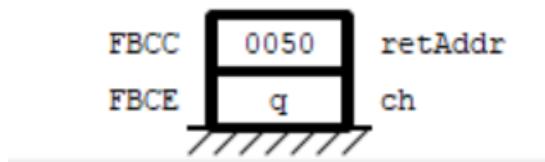
4.c

Run for 3 inputs: one uppercase, one lowercase, & one non-letter and paste a screen shot of each in the Output area of the PEP/8.

The figure consists of four separate windows, each titled "Input/Output".
1. Top-left: Shows the prompt "Please input your character:" followed by "D" and "E".
2. Top-right: Shows the prompt "Please input your character:" followed by "g" and "H".
3. Bottom-left: Shows the prompt "Please input your character:" followed by "4" and "5".
4. Bottom-right: Shows the prompt "Please input your character:" followed by ":" and ";".

4.d

Step thru & Cut and paste the memory trace at a point when in **uppercase** subroutine.



References

Warford, J. (2009). *Computer systems* (4th ed.). Jones and Bartlett.