Homework 2

Brandon Hosley

Mike Davis

Homework 2

Problem 1

A.

82
1. Signed Magnitude: 0101 0010
2. One's Complement: 0101 0010
3. Two's Complement: 0101 0010

B.

-17
1. Signed Magnitude: 1001 0001
2. One's Complement: 1110 1110
3. Two's Complement: 1110 1111

C.

The sum of the Two's Compliment in A and B.
0101 0010 +
1110 1111 =
0100 0001
1. Signed Magnitude: 65
2. One's Complement: 65
3. Two's Complement: 65

Problem 2

Using 14-bit floating point model represent [-100.375]

A.

The number is negative, the sign bit is 1.

B.

$100_{10} = 0110\ 0100_2 => 01.100100 * 2^6$ giving a biased exponent of 0 0110 or corrected (+15) of 1 0101

C.

$100.375_{10} = 01100100.0110_2$
=> 01.10 0100 0110 The significand = 10 0100 0110 (the final two bits will end up being dropped.)

D.

In 14-bit floating point notation: $-100.375_{10} = 1\ 10101\ 1001\ 0001_2$

Problem 3

	Do	Register	Mode	A	X	Mem[0A3C]	Mem[2A42]
Original Content				10B6	FE25	0A41	0A3F
Instruction							
79 2A42	Add to register	Index	Direct	0A3F	2A42	0A41	0A3F
E1 2A42	Store reg to mem	Accumulator	Direct	10B6	2A42	0A41	10B6
A9 0A3C	Bitwise or	Index	Direct	1ABF	FE25	0A41	0A3F
C2 2A42	Load reg from mem	Accumulator	Indirect	xxx	0A3F	0A41	0A3F

Problem 4

A.

Address	Machine Language (Hex)	
0000	C1000C	:Load Register from 000C
0003	18	:Bitwise inversion
0004	F1000B	:Store Byte register to memory too 000B
0007	51000B	:Character Output from 000B
000A	00	:Stop
000B	00	:Empty
000C	FFDA	:1111 1111 1101 1010

B.

This program outputs the character %

C.

The unary instruction at address 0003 is an operation that works on the current value without need of any operands.

D.

This operation inverts the binary in the accumulator:
1111 1111 1101 1010 becomes:
0000 0000 0010 0101 = ASCII %

Problem 5

A.

Algorithm 1 Warford (2009)'s algorithm detailed in *figure 4.31*

Load the machine language program into memory starting at address 0000.
PC ← 0000
SP ← Mem [FFF8]
do
  Fetch the instruction specifier at address in PC
  PC ← PC + 1
  Decode the instruction specifier
  if (*the instruction is not unary*) **then**
    Fetch the operand specifier at address in PC
    PC ← PS + 2
  Execute the instruction fetched
while ((*the stop instruction does not execute*) && (*the instruction is legal*)

For this implementation each instruction is to print an ASCII letter from a specific address.

B.

Address	Machine Language (Hex)	
0000	510016	:Character Output
0003	510017	:Character Output
0006	510018	:Character Output
0009	510019	:Character Output
000C	51001A	:Character Output
000F	51001B	:Character Output
0010	51001C	:Character Output
0013	00	:Stop
0016	42	:ASCII B Character
0017	52	:ASCII R Character
0018	41	:ASCII A Character
0019	4E	:ASCII N Character
001A	44	:ASCII D Character
001B	4F	:ASCII O Character
001C	4E	:ASCII N Character

C.

Plain Hex of the above program:
51 00 16 51 00 17 51 00 18 51 00 19 51 00 1A 51 00 1B 51 00 1C
00 42 52 41 4E 44 4F 4E 00 zz

D.

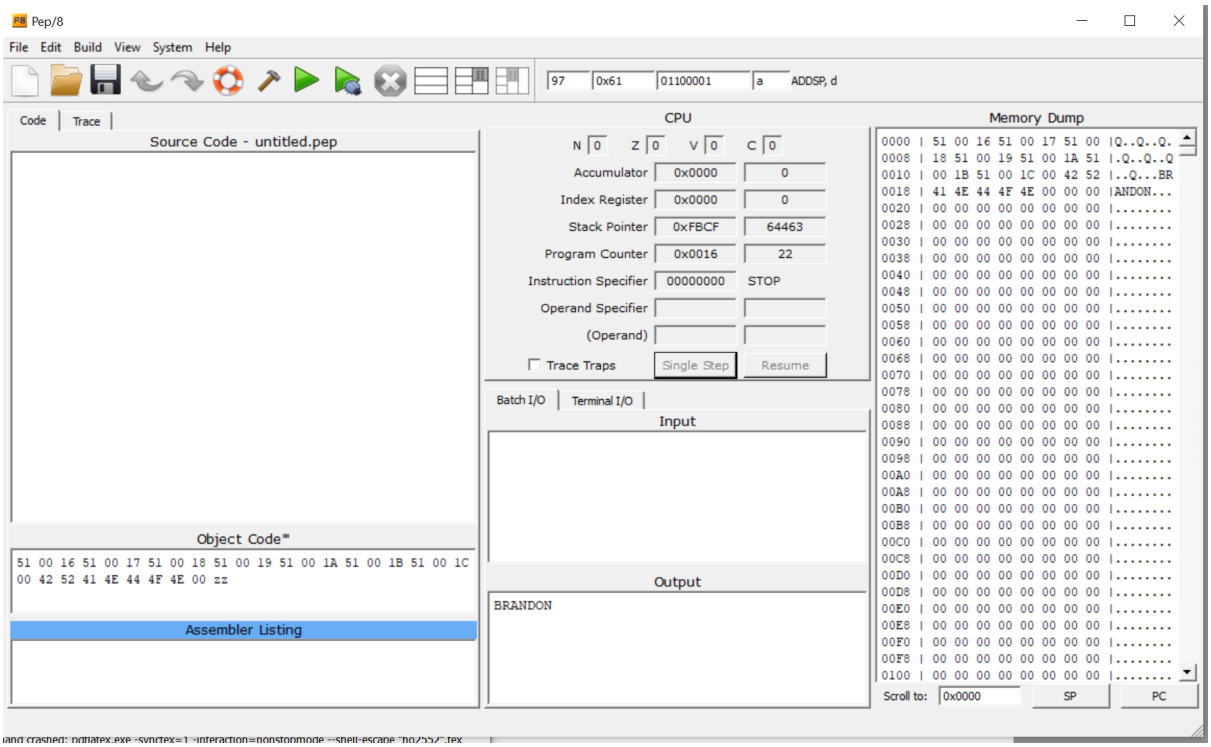

Figure 1. Pep/8 simulator with name output.

Problem 6

A.

Numbers are hardcoded into data at end of program.
Acc ← first number$_{bin}$
Acc += second number$_{bin}$ ▷ in two's compliment form.
Acc += third number$_{bin}$
Acc → *Memory*
Print from Memory

B.

Address	Machine Language (Hex)	
0000	C10015	:Load First Number
0003	710017	:Add Second Number
0006	710019	:Add Third number
0009	A1001B	:Convert to ASCII
000C	F10013	:Save result
000F	510013	:Number Output
0012	00	:Stop
0013	0000	:Result *initially empty*
0015	0006	:First Number (6)
0017	00F9	:Second Number (-7)
0019	0004	:Third Number (4)
001B	0030	:ASCII Mask

C.

Plain hex for above program: C1 00 15 71 00 17 71 00 19 A1 00 1B F1 00 13 51 00 13 00 00 00 00 06 00 F9 00 04 00 30 zz

D.

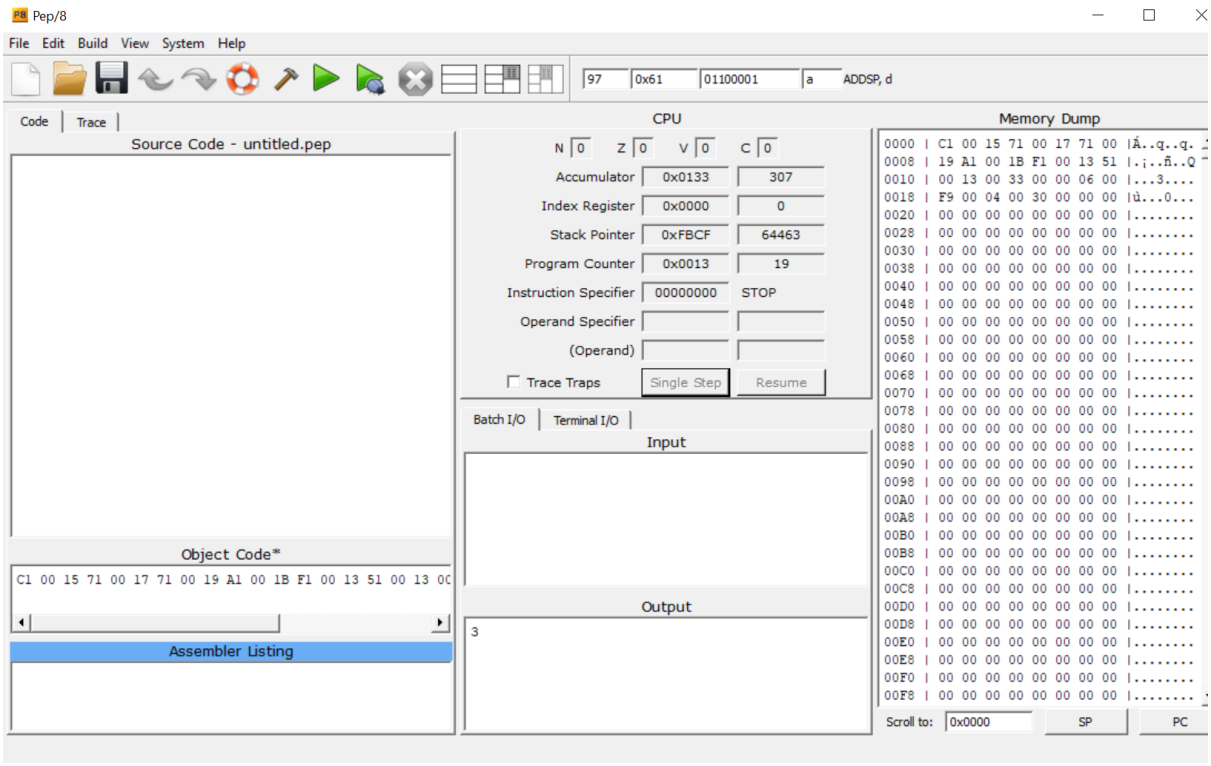

Figure 2. Pep/8 simulator with result of (6)+(-7)+(4) output.

E.

The pep8 would be able to use numbers in the following ranges.
Signed Int: -128 to 127
Unsigned Int: 0 to 255
It should be noted that while all of the operands and the result may fall into this range, if any intermediate results are capable of producing overflow errors.

References

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