JOHN SAN DIEGO COS 331 HW1

This homework is due Friday, September 16th, 5:00 PM. Please remember that no late work will be accepted. You should include the three problems in a single file and submit it to BB before the 5:00 PM deadline.

Problem 1) Please provide a PBRAIN12 program that has equivalent functionality to the following C fragment.

```
int i ;
int Y=10 ;

for(i = 0; i < 10 ; i++)
        y = y + i;
```

ANSWER:

0. P0 = 16	00P016
1. P1 = 17	00P117
2. LOAD ac, 0	030000
3. Store Ac, M (P0) //m[i]	06P0ZZ
4. LOAD Ac, 10	030010
5. Store Ac, M (P1) //m[y]..	06P1ZZ
6. Load Ac, P0	04P0ZZ
7. cmp greater M(16) (27)	270009
8. Jump True Else Label	3315ZZ
9. ACC = p1 + ACC	20P1ZZ
10. Load Ac, P1	06P1ZZ
11. Load P0 , ACC	04P0ZZ
12. ADD ACC, 0001	160001
13. Store M[P0], ACC	06P0ZZ
14. Jump 8	3506ZZ
15. HALT	99ZZZZ

//01 – moves memory location down xx times

10. load pointer to acc 04

11 16 - 160001

PO = ACC – 06P0ZZ

Y =y +i

ACC = p1 + ACC 20p1zz

Problems 2 and 3)

For these problems, assume you have a functional interpreter with all of the VM variables we have discussed (e.g., memory [100] [6], PC, ACC, and P0 - P3 …), and further assume they are globally available. Also, assume the program has been read correctly into the VM memory and that all instructions up to this point have been correctly executed.

Please provide the C code to implement the functions that will execute opcodes 0 and 4. The prototypes for these functions are as follows, where the single parameter is the Instruction Register holding the current instruction:

```
void OP0(char *IR) ;
```

answer:

```
void Op0(char *IR){
int PREG, VAL;
printf("Opcode = 00. Load Pointer Immediate\n");
PrintIR(IR)

PREG = ParseOp1Reg (IR);
VAL = ParseOp2 (IR);

Switch (PREG)
{
        Case0: P0 = VAL;
        Case0: P1 = VAL;
        Case0: P2 = VAL;
        Case0: P3 = VAL;
}

}
```

```
Void OP4 (char *IR);
```

Answer:

```
Void OP4 (char *IR) {
        Int PREG,VAL, ADDR;
        Printf("OPCODE = 04. LOAD Accumulator Register Adressing\n");
        PrintIR();
```

```
          PREG = ParseOp1Reg(IR);

Switch(PREG){
Case 0: ADDR = P0;
Case 1: ADDR = P1;
Case 2: ADDR = P2;
Case 3: ADDR = P3;
}
VAL = (memory[ADDR][2]-48)*1000;
VAL += (memory[ADDR][3]-48)*100;
VAL += (memory[ADDR][4]-48)*10;
VAL += (memory[ADDR][5]-48)*1;
ACC = VAL;
Printf("%i\n",ACC);
}
```