

Project 1A Part I - cq4

d. Change the programs (one at-a-time) to ...

program1.c

```

1 insMEM[0] = 0x10200000; // LOAD R0,(R1) top: R0 = MEM[R1];
2 insMEM[1] = 0x1508FFFF; // ADDI R8, R8, -1 count = count - 1;
3 insMEM[3] = 0x04401000; // ADD R2, R2, R0 sum = sum + R0;
4 insMEM[4] = 0x14210001; // ADDI R1, R1, 1 R1 = R1 + 1;
5 insMEM[5] = 0x2500FFE8; // BNEZ R8 -24 = 6 * 4 = 16 + 8 if (count != 0) goto top
6 insMEM[6] = 0x00000000; // Do not remove. Needed to make HALT work correctly
7 insMEM[9] = 0x28000000; // HALT

```

```

0 1 2 3 4 5 6 7 8 9 10
LOAD R0,(R1) 0IF ID EX ME WB
ADDI R8, R8, -1 1IF ID EX ME WB
NOP 2_ _ _ _ _
ADD R2, R2, R0 3IF ID EX ME WB
ADDI R1, R1, 1 4IF ID EX ME WB
BNEZ R8 -20 5IF ID EX ME WB
NOP 6_ _ _ _ _
NOP 7_ _ _ _ _
NOP 8_ _ _ _ _
HALT 9

```

program2.c

```

1 insMEM[0] = 0x14840001; // ADDI R4, R4, 1 top1: index2++
2 insMEM[3] = 0x10810000; // LOAD R1, 0(R4) element2 = MEM[index2]
3 insMEM[4] = 0x14420001; // ADDI R2, R2, 1 index1++
4 insMEM[7] = 0x10400000; // LOAD R0, 0(R2) element1 = MEM[index1]
5 insMEM[10] = 0x04012800; // ADD R5, R0, R1 temp = element1 + element2
6 insMEM[13] = 0x18C50000; // STORE R5, 0(R6) MEM[index3] = temp
7 insMEM[14] = 0x14C60001; // ADDI R6, R6, 1 index3++
8 insMEM[15] = 0x1508FFFF; // ADDI R8, R8, -1 count--
9 insMEM[18] = 0x2500FFB4; // BNEZ R8 -16*5 + 4 = -76 if (count != 0) goto top
10 insMEM[19] = 0x00000000; // NOP
11 insMEM[22] = 0x28000000; // HALT

```

```

0 1 2 3 4 5
ADDI 0 IF ID EX MEM WB // ADDI R4, R4, 1
NOP 1_ _ _ _ _
NOP 2_ _ _ _ _
Load 3 IF ID EX MEM WB // LOAD R1, 0(R4)
ADDI 4 IF ID EX MEM WB // ADDI R2, R2, 1
NOP 5_ _ _ _ _
NOP 6_ _ _ _ _
Load 7 IF ID EX MEM WB // LOAD R0, 0(R2)
NOP 8_ _ _ _ _
NOP 9_ _ _ _ _
ADD 10 IF ID EX MEM WB // ADD R5, R0, R1
NOP 11_ _ _ _ _
NOP 12_ _ _ _ _
STORE R5, 0(R6) 13 IF ID EX MEM WB
ADDI R6, R6, 1 14 IF ID EX MEM WB
ADDI R8, R8, -1 15 IF ID EX MEM WB
NOP 16
NOP 17
BNEZ R8 -32 + 4 = -36 18 IF ID EX MEM WB // 18 * 4 = 72
NOP 19
NOP 20
NOP 21
HALT 22

```

program3.c

```

1 insMEM[0] = 0x10200000; // LD R0, 0(R1) COPY: value = MEM[index]
2 insMEM[3] = 0x18200200; // STORE R0, 512(R1) MEM[index+512] = value
3 insMEM[4] = 0x1508FFFF; // ADDI R8, R8, -1 count--
4 insMEM[5] = 0x14210001; // ADDI R1, R1, 1 index++
5 insMEM[7] = 0x2500FFE0; // BNEZ R8 -20 (7+1)*4 = 32 if (count != 0)
goto COPY
6 insMEM[8] = 0x00000000; // NOP
7 insMEM[11] = 0x28000000; // HALT

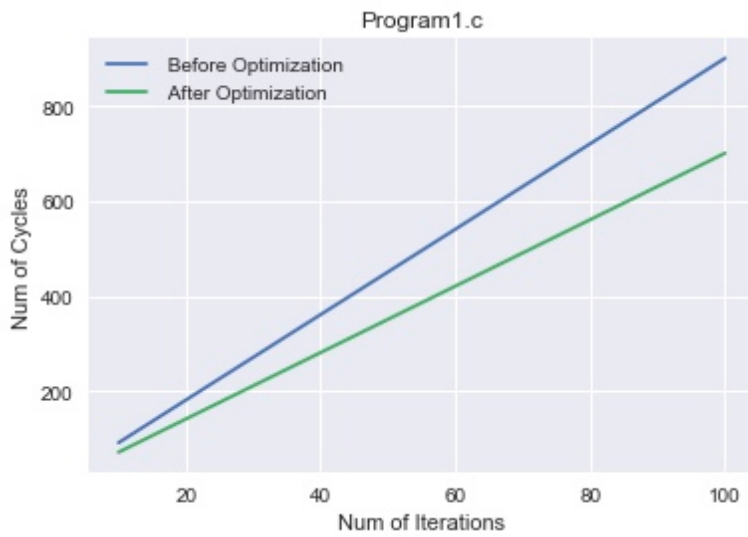
```

```

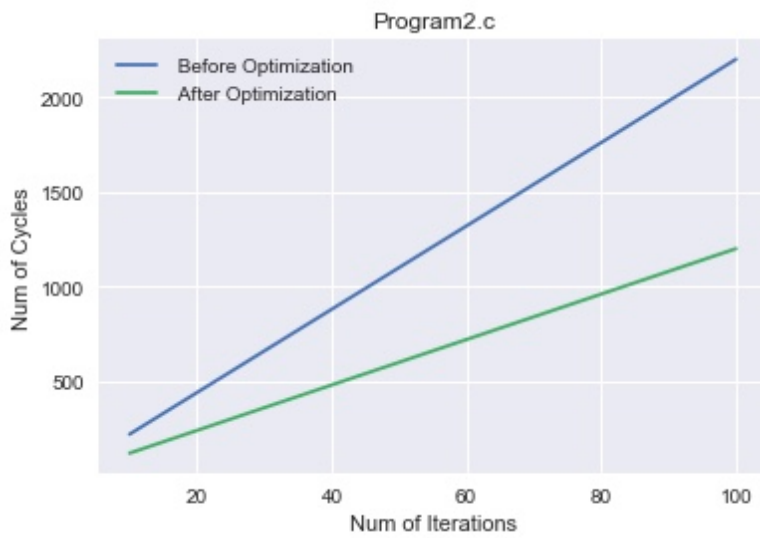
LD R0, 0(R1) 0 IF ID EX ME WB
NOP 1_ _ _ | _
NOP 2_ _ | _ _
STORE R0, 512(R1) 3 IF ID EX ME WB
ADDI R8, R8, -1 4 IF ID EX ME WB
ADDI R1, R1, 1 5 IF ID EX ME WB
NOP |
BNEZ R8 -20 7 IF ID EX ME WB
NOP 8
NOP 9
NOP 10
HALT 11

```

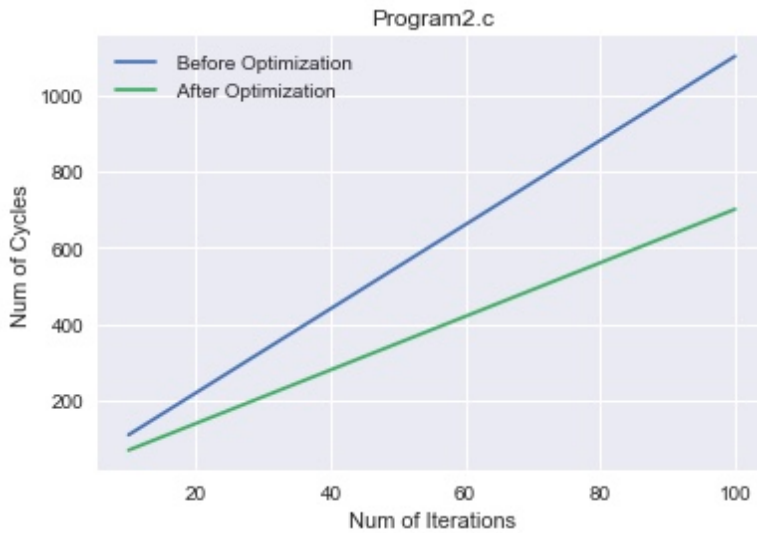
e. Run each program for NUM_ITERATIONS equal ...



```
x = [10, 20, 40, 80, 100]
y = [91, 181, 361, 721, 901]
y2 = [71, 141, 281, 561, 701]
```



```
x = [10, 20, 40, 80, 100]
y = [221, 441, 881, 1761, 2201]
y2 = [121, 241, 481, 961, 1201]
```



```
x = [ 10, 20, 40, 80, 100]
y = [111, 221, 441, 881, 1101]
y2 = [ 71, 141, 281, 561, 701]
```

f. Rewrite the code of each program to minimize ...

program1.c

```
1 insMEM[0] = 0x1508FFFF; // ADDI R8, R8, -1      count = count - 1;
2 insMEM[1] = 0x10200000; // LOAD R0,(R1)          top: R0 = MEM[R1];
3 insMEM[2] = 0x14210001; // ADDI R1, R1, 1          R1 = R1 + 1;
4 insMEM[3] = 0x2500FFF0; // BNEZ R8 -20           if (count != 0) goto top
5 insMEM[4] = 0X04401000; // ADD R2, R2, R0          sum = sum + R0;
6 insMEM[5] = 0x00000000; // Do not remove. Needed to make HALT work correctly
7 insMEM[7] = 0x28000000; // HALT
```

After optimization

```

      |   |   |   |   |   |   |   |   |   |   |   |
      0  1  2  3  4  5  6  7  8  9 10
ADDI  R8, R8, -1  0IF ID EX ME WB
LOAD  R0,(R1)    1IF ID EX ME WB
ADDI  R1, R1, 1   2IF ID EX ME WB
BNEZ  R8 -20     3IF ID EX ME WB
ADD   R2, R2, R0  4IF ID EX ME WB
NOP                                5_ _ _ _ _
NOP                                6_ _ _ _ _
HALT  7
```

After optimization, each loop there're only 7 steps.

program2.c

```
1 insMEM[0] = 0x14840001; // ADDI R4, R4, 1  top1: index2++
2 insMEM[1] = 0x14420001; // ADDI R2, R2, 1    index1++
3 insMEM[2] = 0x1508FFFF; // ADDI R8, R8, -1    count--
4 insMEM[3] = 0x10810000; // LOAD R1, 0(R4)     element2 = MEM[index2]
5 insMEM[4] = 0x10400000; // LOAD R0, 0(R2)     element1 = MEM[index1]
```

```

6 insMEM[7] = 0x04012800; // ADD R5, R0, R1          temp = element1 + element2
7 insMEM[8] = 0x2500FFDC; // BNEZ R8 -16*2 + 4 = -36 = 9 * 4      if (count != 0) goto
  top
8 insMEM[9] = 0x14C60001; // ADDI R6, R6, 1          index3++
9 insMEM[10] = 0x18C50000; // STORE R5, 0(R6)        MEM[index3] = temp
10 insMEM[11] = 0x00000000; // NOP
11 insMEM[12] = 0x28000000; // HALT

```

```

  0  1  2  3  4  5
ADDI 0IF ID EX ME WB          // ADDI R4, R4, 1
ADDI 1IF ID EX ME WB          // ADDI R2, R2, 1
ADDI 2IF ID EX ME WB          // R8, R8, -1
Load 3IF ID EX ME WB          // LOAD R1, 0(R4)
Load 4IF ID EX ME WB          // LOAD R0, 0(R2)
NOP 5_ _ _ _ _
NOP 6_ _ _ _ _
ADD 7IF ID EX ME WB          // ADD R5, R0, R1
BNEZ R8 -32 + 4 = -36      8IF ID EX ME WB      // 18 * 4 = 72 = 16*6
ADDI R6, R6, 1          9IF ID EX ME WB
STORE R5, 0(R6)        10IF ID EX ME WB
HALT                   11

```

program3.c

```

1 insMEM[0] = 0x1508FFFF; // ADDI R8, R8, -1          count--
2 insMEM[1] = 0x10200000; // LD R0, 0(R1)          COPY: value = MEM[index]
3 insMEM[2] = 0x14210001; // ADDI R1, R1, 1          index++
4 insMEM[3] = 0x2500FFF0; // BNEZ R8 -20 (3+1)*4 = 16 if (count != 0) goto COPY
5 insMEM[4] = 0x18200200; // STORE R0, 512(R1)      MEM[index+512] = value
6 insMEM[7] = 0x28000000; // HALT

```

```

  0  1  2  3  4  5
ADDI R8, R8, -1      0IF ID EX ME WB
LD R0, 0(R1)        1IF ID EX ME WB
ADDI R1, R1, 1      2IF ID EX ME WB
BNEZ R8 -20          3IF ID EX ME WB
STORE R0, 512(R1)    4IF ID EX ME WB
NOP 5
NOP 6
HALT 7

```