1.

3.4.1：输出8，d是局部变量

3.4.2：

输出：

10

10

3.4.3：a是全局变量，b是局部变量，输出：11

3.4.4：

执行do\_add(a, b)时，将局部变量c修改为a+b = 5，再次调用函数do\_sub(a, b)修改局部变量c，里面的a和b分别为c(为5)和1，局部变量c = a-b = 5 - 1 = 4， 再用do\_mul(a, b)修改局部变量c,调用全局变量c，将其修改为4 \* 4 = 16, 打印16 并返回16，局部变量c变为16，再执行局部变量c = do\_div(c, 2)，函数里的a=16, b=2,里面的局部变量c = a/b = 8.0,打印8.0，返回8.0，在do\_sub(a, b)里局部变量c=8.0，打印8.0，返回8.0，局部变量c=8.0，打印8.0，最后再次打印全局变量c，即16

输出:

16

8.0

8.0

8.0

16

3.4.5：a是函数里面的变量，不能全局化

3.4.6：函数中那个a=a+10操作将那个a默认为局部变量，但函数里没有定义这个a

2. k=0

3.14：40 40

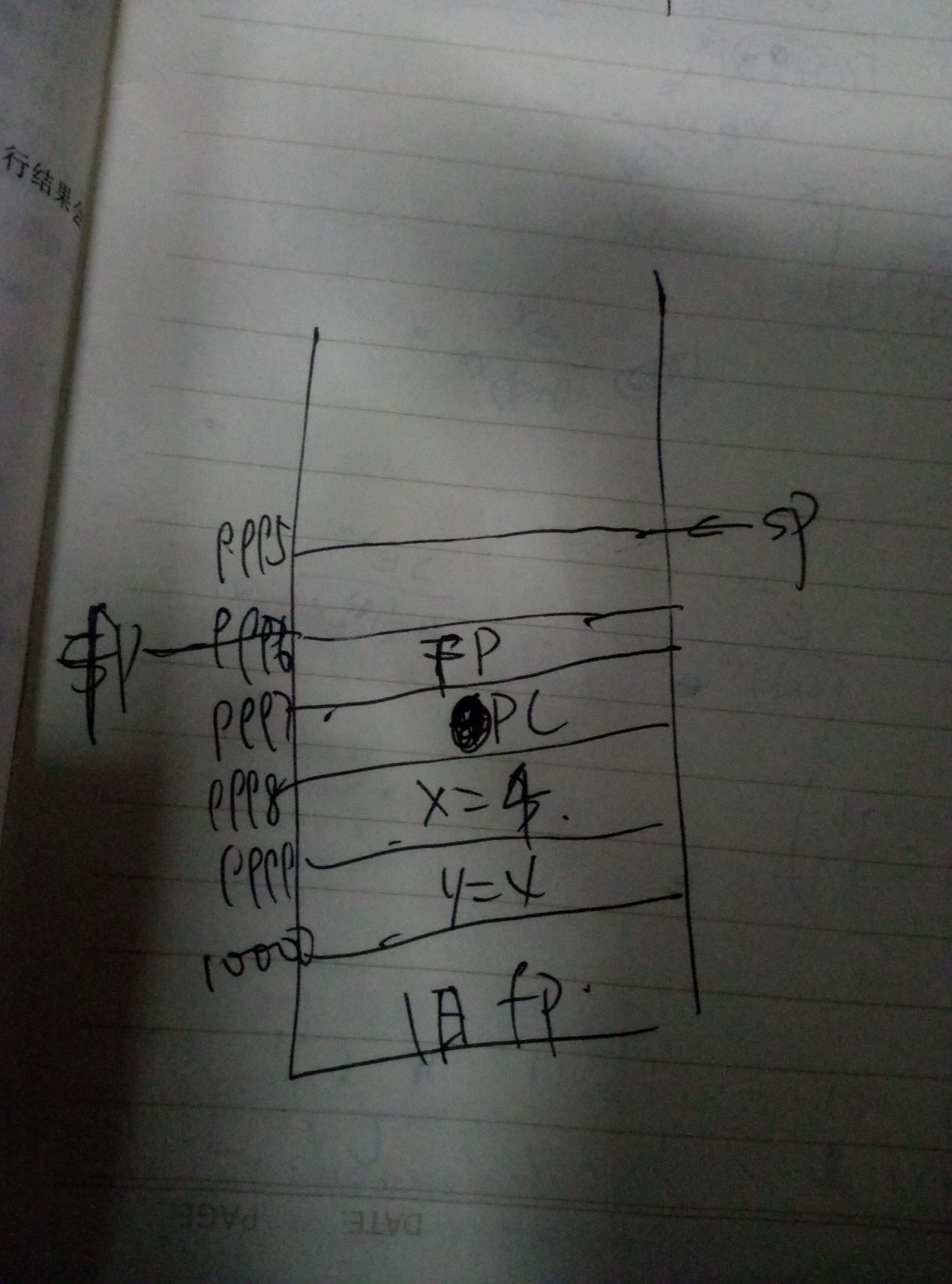
3.16：

20

30 15

3.18：120

3.20：



3.5.1

Factor: 2

Factor: 3

Prime Factor: 3

局部变量：参数x:3，变量y:1

3.

move R15,10000 #R15表示fp，fp = 10000

move sp,R15 #sp = fp

sub sp,sp,2 #sp从10000往下开辟3个空间给局部变量x，y，sp = sp-2

move R2,5 #x = 5

move R3,6 #y = 6

store -1(R15),R3

store -2(R15),R2

push R3 #传参数b

push R2 #传参数a

call Lf #调用函数f(a,b),返回值存在R1中

goto Lprint

#f函数有两个参数a，b，将a+b-1放到R1中返回

Lf: #f(a,b)

push R15 #将旧的fp值压入栈内

move R15,sp #新的fp = sp

sub sp,sp,1 #留一个空间，存放局部变量c

push R2 #在函数中被更改，所以先存入栈内，在return之前会pop出来这个值

push R3 #在函数中被更改，所以先存入栈内，在return之前会pop出来这个值

load R2,2(R15) #R2 = a

load R3,3(R15) #R3 = b

add R1,R2,R3

sub R1,R1,1

store -1(R15),R1 #存放 c

Lreturn:

pop R3 #返回初始的R3

pop R2 #返回初始的R2

move sp,R15 #sp = fp

pop R15 #重置fp，成为旧的fp

ret

Lprint:

\_pr R1



4.

#def factors(x): #找到x的因数

# y=x//2

# for i in range(2,y+1):

# if (x %i ==0): #发现i是x的因数

# print("Factor:",i);

# factors(x//i) #递归调用自己，参数变小是x//i

# break #跳出for循环

# else: #假如离开循环正常，没有碰到break，就执行else内的print，x是质数

# print("Prime Factor:",x)

# print("参数x:%d, 变量y:%d" %(x,y))

# return

#factors(18)

move R15,10000

move sp,R15

move R2,18

push R2

call Lf

goto Lend

Lf:

push R15

move R15,sp

sub sp,sp,2

move R4,2

div R3,R2,2

store -1(R15),R3

Lfor:

slt R6,R4,R3

beqz R6,Lelse

div R8,R2,R4

mul R7,R8,R4

sub R7,R7,R2

beqz R7,Ldigui

add R4,R4,1

goto Lfor

Ldigui:

\_pr 'factor',R4

store -2(R15),R4

move R2,R8

push R2

call Lf

goto Lprint

Lelse:

\_pr 'prime factor:',R2

goto Lprint

Lprint:

load R2,2(R15)

load R3,-1(R15)

\_pr 'x:',R2,' y:',R3

Lreturn:

move sp,R15

pop R15

ret

Lend:



5.

#s = '101'

#l = 3

#p = 2\*\*(l-1)

#q = 0

#for each in s:

# if int(each):

# q +=p

# p //= 2

#print(q)

\_data 0, [3, 49, 48, 49]

load R0,0(0)

move R2,1

move R3,0 #答案

move R15,0

move R13,0

shiftl R2,R2,R0

Lfor:

slt R4,R15,R0

sub R0,R0,1

beqz R4,L0

goto L1

L1:

load R1,1(R13)

add R13,R13,1

sle R4,R1,48

shiftr R2,R2,1

beqz R4,Lplus

goto Lfor

Lplus:

add R3,R3,R2

goto Lfor

L0:

\_pr R3



6.

#L = [9, 12, 1, 100, 31, 5, 6, 29, 90, 10, 51, 21, 41, 23, 56, 64, 90, 98, 73, 73]

#p = 20

#for j in range(p-1,-1,-1):

# for i in range(j):

# if L[i+1]<L[i]:

# L[i+1],L[i] = L[i],L[i+1]

#print(L)

\_data 10, [20, 9, 12, 1, 100, 31, 5, 6, 29, 90, 10, 51, 21, 41, 23, 56, 64, 90, 98, 73, 73]

load R0,0(10) # R0 = p

move R2,1

move R5,29 # R5 = j

move R6,10

move R15,0

move R14,R0

Lfor1:

move R3,9 # R3 = i

sle R4,R5,R6 #if j<=10

sub R5,R5,1

beqz R4,Lfor2

goto Lprint

Lfor2:

slt R4,R3,R5 # if i<j

add R3,R3,1

beqz R4,Lfor1

goto Lif

Lif:

load R7,2(R3)

load R8,1(R3)

slt R4,R7,R8 #if L[i+1]<L[i]

beqz R4,Lfor2

goto Lchange

Lchange: # L[i+1],L[i] = L[i],L[i+1]

store 1(R3),R7

store 2(R3),R8

goto Lfor2

Lprint:

\_pr 1(10),2(10),3(10),4(10),5(10),6(10),7(10),8(10),9(10),10(10),11(10),12(10),13(10),14(10),15(10),16(10),17(10),18(10),19(10),20(10)

