1. コンピュータの数値表現に関する以下の問いに答えよ。

(1) 10 進数の 19 を 6bit の 2 進数表現にせよ。

010011

(2) (1) の数について 2 の補数を求めよ。これを 8bit に拡張せよ。さらに 16 進数で示せ。

101101　11101101　ED

(3) (2) で示した 2 の補数を用いることで、10 進数 35－19 を計算せよ。経過も示すこと。

35 は 100011　100011＋101101＝1001000　最上位は無視

2. 16bit RISC POCO で以下の命令を順に実行した。

LDI r2,#2　　01000 010 00000010　　r2 ←2

LD r1,(r2)　　00000 001 010 01001　　r1←1

LD r2,(r1)　　00000 010 001 01001　　r2←2

ADD r1,r2　　00000 001 010 00110　　r1←3

ここで、データメモリの中身は 0 番地 3、1 番地 2、2 番地 1、3 番地 0 に初期化されている。

(1) 上記の命令の機械語表現を示せ。(右)

(2) r1,r2 の変化を示せ。

3. 16bit RISC POCO でメモリの 0 番地から格納されている 100 個の数の総和を求めて答えを 100 番地に格納するプログラムを書け。

```
    LDI r1,#100
    LDI r2,#0
loop: ADDI r1,#-1
      LD r3,(r1)
      ADD r2,r3
      BNZ r1,loop
      LDI r1,#100
      SW r2,(r1)
end:  JMP end
```

4. 3 のプログラムをメモリの X 番地から格納する Y 個の数の総和を求めるサブルーチンとする。X は r0, Y は r1 に与え、答えは r3 に戻す。このサブルーチンのプログラムを書け。

```
      LDI r3,#0
loop: LD r2,(r0)
      ADD r3,r2
      ADDI r0,#1
      ADDI r1,# -1
      BNZ r1,loop
      BR r7
```

5. 4 のプログラムでスタックにセーブしなければならないレジスタはどれかを示せ。

r0,r1,r2

6. 64Kword の主記憶に対して 1Kword のキャッシュを作る。それぞれの方式でタグメモリの構成がどのようになるかを示せ。ただし、1 ブロックは 32word とする。

(1) ダイレクトマップ (2) 2-way セットアソシアティブキャッシュ

ダイレクトマップ

キャッシュには 32 ブロック入る　index=32, block 内アドレスは 5 ビットなので、tag=6bit

6 ビット幅、深さ 32 のタグ×1

2-way では

7 ビット幅、深さ 16 のタグ×2

7. ダイレクトマップのライトバックキャッシュにおいて互いにコンフリクトミスを起こす番地 A と B に対して順に以下のアクセスを行った。アクセスがヒット（H）するかミス（M）するか、アクセスによってライトバック（W）とリプレイス（R）が起きるか、アクセスの結果、ブロックは Dirty(D)になるか Clean(C)になるかを示せ。

(1) A から読み出し　(2) A に書き込み　(3) B から読み出し　(4) B から読み出し　(5) A に書き込み

(1)　M、R、C
(2)　H、D
(3)　M、W+R、C
(4)　D、C
(5)　M,　W+R、D

8. 遅延スロット 1 の遅延分岐を利用するパイプライン CPU について、分岐命令の確率を 20%、遅延スロットを有効に埋めることができない確率を 5%とした場合の CPI を計算せよ。ただし理想の CPI は 1 とする。

１＋0.2×0.05＝1.001

9. 割り込みが必要な理由について簡単に説明せよ。

割り込みにより要求を伝えることができないと、CPU は常に I/O の状態をビジーウェイトする必要が生じて、処理が進まず、プログラムも困難だから。

10. パイプライン処理がマルチコアなどの並列処理よりも優れている点を簡単に説明せよ。

○プログラムを並列化しなくても性能を改善できる

○それぞれのステージは CPU の資源を部分的に分かち持てばよいためコストが小さい。

など。

参考　POCO の代表的な命令

MV rd,rs	rd← rs	**00000dddsss00001**
AND rd,rs	rd← rd AND rs	**00000dddsss00010**
OR rd,rs	rd← rd OR rs	**00000dddsss00011**
ADD rd,rs	rd← rd + rs	**00000dddsss00110**
SUB rd,rs	rd← rd - rs	**00000dddsss00111**
ST rd,(ra)	(ra)← rd	**00000dddaaa01000**
LD rd,(ra)	rd← (ra)	**00000dddaaa01001**
JR rd	pc ←　rd	**00000ddd---01010**
LDI rd,#X	rd← X(符号拡張)	**01000dddXXXXXXXX**
LDIU rd,rs	rd← X(ゼロ拡張)	**01001dddXXXXXXXX**
ADDI rd,#X	rd←rd+X(符号拡張)	**01100dddXXXXXXXX**
ADDIU rd,#X	rd←rd+X(ゼロ拡張)	**01101dddXXXXXXXX**
LDHI rd,#X	rd←{X,0}	**01010dddXXXXXXXX**
BEZ rd,X	if(rd=0) pc←pc+X+1	**10000dddXXXXXXXX**
BNZ rd,X	if(rd≠0) pc←pc+X+1	**10001dddXXXXXXXX**
BPL rd,X	if(rd>=0) pc←pc+X+1	**10010dddXXXXXXXX**
BMI rd,X	if(rd<0) pc←pc+X+1	**10011dddXXXXXXXX**

JAL #X	pc←pc+X+1, r7←pc+1	10101XXXXXXXXXXX
JMP #X	pc←pc+X+1	10100XXXXXXXXXXX