問題中に不明な点がある場合や、手元に資料がない場合は、各自判断して答え、その旨を記してください。

1. 0x1010 番地に A、0x1011 番地に B が入っている。(A+B) AND (A<<1)を計算し、答を 0x1012 番地に格納する POCO のプログラムをアセンブリ言語で記述せよ。(ヒント LDHI 命令を使う)

答

```
LDHI r0,#0x10          01010_000_00010000
ADDIU r0,#0x10       01101_000_00010000
LD r1,(r0)               00000_001_000_01001
ADDIU r0,#1
LD r2,(r0)
ADD r2,r1
SL r1
AND r2,r1
ADDIU r0,#1
ST r2,(r0)
```

LDHI r0,#0x1010 というのは不可能。これは 2 でコーディングできないという事実に気づいて欲しかった。部分点は上げている。

2. 問題 1 のプログラムの最初の 3 行を機械語に変換せよ。(上)

3. stop 入力を 1 にすると、偶数の目で停止するイカサマサイコロを Verilog HDL で記述せよ。入力は stop およびクロック信号 clk、リセット信号 rst_n (0 で cnt が 1 となるとし、出力は 3 ビットの目を表すカウンタ出力 cnt とせよ。

```
module ikasama(input clk,rst_n,stop, output [2:0] cnt);
    always @(posedge clk or negedge rst_n) begin
        if(!rst_n) cnt <= 1;
        else if(stop) cnt <= cnt & 3'b110;
        else if(cnt==6) cnt <= 1;
        else cnt <= cnt+1;
  end
endmodule
```

これは一例で、目が偶数にしか変わらないサイコロも正解にしたが、本当は、それはちょっとまずい。止まった時だけ偶数になるのが正しい。2 でしか止まらないサイコロは、いかさまがすぐばれるので、減点した。

4. 0 番地から 15 番地までの 16 個の符号付き数のうち、負の数がいくつあるかを数えて、答えを 0x10 番地に書き込むプログラムを書け。

```
      LDI r0,16
      LDI r1,#0
Loop: ADDIU r0, #-1
       LD r2,(r0)
        BPL r2, Skip
          ADD r1,#1
Skip: BEZ r0, Loop
       LDI r0,#0x10
       ST r1,(r0)
```

#0x8000 をどこかのレジスタに入れて AND する方法もあるが、BPL で符号ビットを調べる方が楽。カウンタを使ってもいいし、r0 のカウントダウンを最後に持ってきて BPL で Loop に飛ばす手もある。

5. 除算サブルーチン div は、r4/r5 を計算し商を r2、余りを r3 に格納する。この除算サブルーチンを利用し、0 番地から 15 番地までの 16 個のデータのうち 3 の倍数がいくつあるかを数えて、その答えを 0x10 番地に書き込むプログラムを書け。

```
      LDI r0,16
      LDI r1,#0
      LDI r5,#3
Loop: ADDIU r0, #-1
       LD r4,(r0)
      JAL div
        BPL r3, Skip
          ADD r1,#1
Skip: BEZ r0, Loop
       LDI r0,#0x10
       ST r1,(r0)
```

プログラムの構造は 4 をそのまま利用できる。r4 にメモリからの値を持ってこなければサブルーチンが使えない。

6. 全ての命令を 2 サイクルで実行する 2 サイクル版 POCO2 を目標周期 5nsec で論理合成したところ、slack が 0.1nsec となった。一方、全ての命令を 3 サイクルで実行する POCO3 を目標周期 4nsec で論理合成したところ、slack が-0.1nsec となった。どちらが、どれだけ高速かを求めよ。

   2 サイクル版　2×4.9×命令数＝9.8×命令数

   3 サイクル版　3×4.1×命令数＝12.3×命令数

   12.3／9.8＝1.255　　2 サイクル版が 1.26 倍速い

   9.8 と 12.3 まで正解で、明らかに最後のみ計算ミスの人には部分点を挙げているが、電卓持ち込み可なのでちゃんと計算してほしい。

7. LDIL rd,#X 命令は、イミーディエイト部分の 8 ビット X をレジスタ rd の下位 8 ビットに書き込むが、rd の上位 8 ビットは元の値を維持する命令である。添付の POCO の Verilog 記述をどのように変更すれば良いかを示せ。def.h には LDIL が定義されているとする。

```
wire ldil_op;
assign ldil_op = (opcode == `OP_LDIL);
assign alu_b = (addi_op | ldi_op) ? {{8{imm[7]}},imm}:
                    (addiu_op | ldiu_op)?{8'b0,imm}:
                     ldil ? {rf_a[15:8],imm}: rf_b;
assign com = … (ldi_op | ldiu_op | ldil_op): `ALU_THB: ….
assign rwe = …. | ldiu_op;
….
```

修正点のみ示した。赤字の所がポイントで、ここができないと本来ダメなのだが部分点は上げた。他にも ALU の A 入力からまわす方法、rf_c から直接入れる方法もある。

8. 4M ワードの主記憶に対して 64K ワードのキャッシュを設ける。ブロックサイズを 64 ワードとした時、（1）ダイレクトマップキャッシュのディレクトリ構成、（2）4-way セットアソシアティブキャッシュのディレクトリ構成をそれぞれ述べよ。

（1）　ダイレクトマップのキャッシュ中には 64K/64=1K ブロック存在する。よって index=10bit。4M ワードの主記憶は 22bit、ブロック内アドレスは 6bit、したがってタグ＝22－6－10＝6bit、幅 6 ビット深さ 1 K のテーブルが必要

（2）　4way になるので、index は 10－2 は 8bit、タグは 6+2=8bit、幅 8 ビット、深さ 256 のテーブルが 4 個必要

これはほとんどの人が正解だった。これを落とすと辛い。

9. TLB(Translation Lookaside Buffer)は、ダイレクトマップ方式や 2way セットアソシアティブ方式よりもフルアソシアティブ方式を用いることが多い。この理由を簡単に述べよ。

   局所性が高いのでヒット率は高く、容量ミスは少ない。しかし衝突ミス時の可能性があり、その時の性能低下が大きい。一方で高速アクセスが必要。このため、エントリ数が少なくてヒット率が高いフルアソシアティブ方式が適している。

10. あるプログラムについて並列化可能な部分 r については、強力なアクセラレータを用いると 100 倍高速化できる。プログラム全体の性能を 50 倍高速化するには r が何%以上あれば良いか？

    0.02＝（1－r）＋r／100

    0.02-1.0 = -0.99r

    -0.98=0.99r =98.99%　結構高い割合でないとまずいことが分かる。99%でいいことにした。この問題は中学生でも解けるので正解して欲しい。