# 情報工学科 計算機構成 2013 年 期末試験 (担当:天野 持ち込み何でも可)

**POCO の命令セット、ハードウェア構成は 2 ページ目以降に示しますが、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、その旨を記して下さい。注意：問題用紙は 4 ページあります。**

1. 0 番地に A, 1 番地に B が入っている。(A-B)AND(A+B) を計算し、答えをレジスタ 3 に入れる POCO のプログラムをアセンブリ言語で記述せよ。
2. 問題 1 のプログラムの最初の 4 行を機械語に変換せよ。
3. 8bit の入力 A,B と選択入力 S を持ち、S=0 の時は A+B を、S=1 の時は A-B を計算し、Y に出力するモジュールを Verilog HDL で記述せよ。
4. マルチサイクル CPU のシングルサイクル CPU に対する利点を簡単に説明せよ。
5. 論理合成を行う場合、目標周期を小さくしすぎて slack がマイナスになった場合、どのような問題が生じるかを簡単に説明せよ。
6. 下のコードはサブルーチンの出口で、レジスタを復帰している部分である。これに対応した、レジスタを退避するコードを書け。

```
...
 LD r7,(r6)
 ADDI r6,#1
 LD r1,(r6)
 ADDI r6,#1
 JR r7
```

7. 0 番地から 8 つの正の整数が並んでいる。このうち奇数の個数を調べるプログラムを書け。結果は r6 に入れよ。
8. 添付の POCO の Verilog 記述と図 1 のデータパスが食い違っている点を一つ述べよ。(たくさんあるのでどれでもよい)
9. 64K ワードのアドレス空間に対して 4K ワードのキャッシュを設ける。ブロックアドレスを 16 ワードとした時、4way set associative cache のキャッシュディレクトリ (タグメモリ) の構成を示せ。

10. あるスーパーコンピュータは 1000 プロセッサを持っていて、プログラムの並列化可能な部分については 1 プロセッサで実行する場合の 1000 倍性能が上がる。並列化が 99%可能だが、残りは 1 プロセッサで実行できないプログラムの性能は、何倍に上がるかを計算せよ。

全 10 点

A) POCO の命令コード

```
NOP                                          00000 --- --- 00000
MV  rd,rs    rd <- rs                        00000 ddd sss 00001
AND rd,rs    rd <- rd AND rs                 00000 ddd sss 00010
OR  rd,rs    rd <- rd OR rs                  00000 ddd sss 00011
SL  rd       rd <- rd<<1                     00000 ddd --- 00100
SR  rd       rd <- rd>>1                     00000 ddd --- 00101
ADD rd,rs    rd <- rd + rs                   00000 ddd sss 00110
SUB rd,rs    rd <- rd - rs                   00000 ddd sss 00111
ST rs, (ra)     rs -> (ra)                   00000 sss aaa 01000
LD rd, (ra)     rd <- (ra)                   00000 ddd aaa 01001
LDI rd,#X       rd <- X (符号拡張)           01000 ddd XXXXXXXX
LDIU rd,#X      rd <- X (符号拡張なし)       01001 ddd XXXXXXXX
ADDI rd,#X      rd <- rd + X (符号拡張)      01100 ddd XXXXXXXX
ADDIU rd,#X     rd <- rd + X (符号拡張なし)  01101 ddd XXXXXXXX
LDHI rd,#X      rd <- X|0                    01010 ddd XXXXXXXX
BEZ rd, X   if (rd==0) pc <- pc + X          10000 ddd XXXXXXXX
BNZ rd, X   if (rd!=0) pc <- pc + X          10001 ddd XXXXXXXX
BPL rd, X   if (rd>=0) pc <- pc + X          10010 ddd XXXXXXXX
BMI rd, X   if (rd<0) pc <- pc + X           10011 ddd XXXXXXXX
JMP X       pc <- PC + X                     10100 XXXXXXXXXXX
JAL X       r7 <- pc, pc<- pc + X            10101 XXXXXXXXXXX
JR  rd                pc <- rd               00000 ddd --- 01010
```

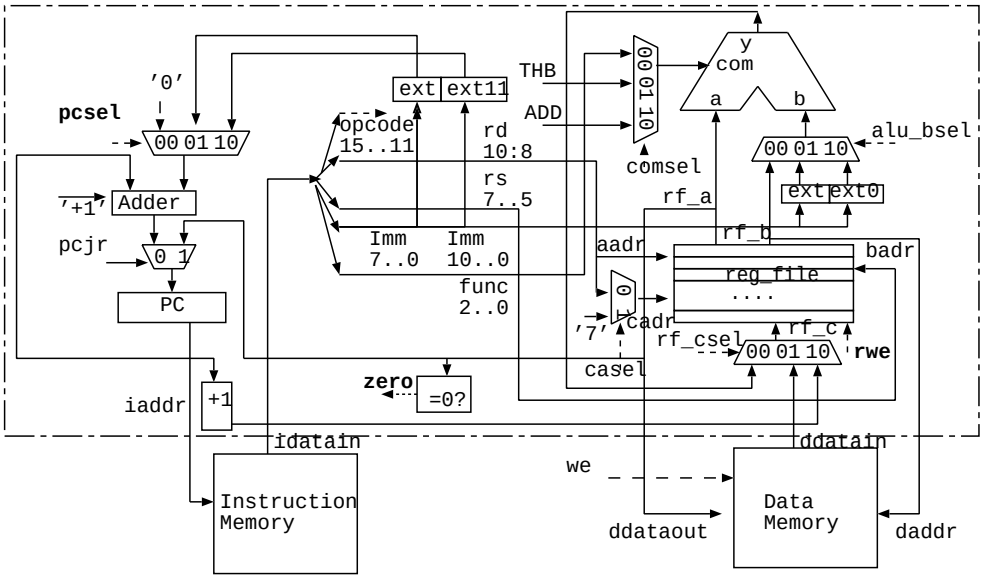


図 1: POCO のデータパス

```
‘include "def.h"
module poco(
input clk, rst_n,
input [‘DATA_W-1:0] idatain,
input [‘DATA_W-1:0] ddatain,
output [‘DATA_W-1:0] iaddr, daddr,
output [‘DATA_W-1:0] ddataout,
output we);

reg [‘DATA_W-1:0] pc;
wire [‘DATA_W-1:0] rf_a, rf_b, rf_c;
wire [‘DATA_W-1:0] alu_b, alu_y;
wire [‘OPCODE_W-1:0] opcode;
wire [‘OPCODE_W-1:0] func;
wire [‘REG_W-1:0] rs, rd;
wire [‘SEL_W-1:0] com;
wire [‘IMM_W-1:0] imm;
wire rwe;
wire st_op, bez_op, bnz_op, addi_op, ld_op, alu_op;
wire ldi_op, ldiu_op, addiu_op;

assign ddataout = rf_a;
assign iaddr = pc;
assign daddr = rf_b;

assign {opcode, rd, rs, func} = idatain;
assign imm = idatain[‘IMM_W-1:0];

// Decorder
assign st_op = (opcode == ‘OP_REG) & (func == ‘F_ST);
assign ld_op = (opcode == ‘OP_REG) & (func == ‘F_LD);
assign alu_op = (opcode == ‘OP_REG) & (func[4:3] == 2’b00);
assign ldi_op = (opcode == ‘OP_LDI);
assign ldiu_op = (opcode == ‘OP_LDIU);
assign addi_op = (opcode == ‘OP_ADDI);
assign addiu_op = (opcode == ‘OP_ADDIU);
assign bez_op = (opcode == ‘OP_BEZ);
assign bnz_op = (opcode == ‘OP_BNZ);

assign we = st_op;

assign alu_b = (addi_op | ldi_op) ? {{8{imm[7]}},imm} :
(addiu_op | ldiu_op) ? {8’b0,imm} : rf_b;
```

```
assign com = (addi_op | addiu_op ) ? `ALU_ADD:
(ldi_op | ldiu_op ) ? `ALU_THB: func[`SEL_W-1:0];

assign rf_c = ld_op  ? ddatain : alu_y;
assign rwe = ld_op  | alu_op | ldi_op | ldiu_op | addi_op | addiu_op  ;

alu alu_1(.a(rf_a), .b(alu_b), .s(com), .y(alu_y));

rfile rfile_1(.clk(clk), .a(rf_a), .aadr(rd), .b(rf_b), .badr(rs),
.c(rf_c), .cadr(rd), .we(rwe));

always @(posedge clk or negedge rst_n)
begin
   if(!rst_n) pc <= 0;
   else if ((bez_op & rf_a == 16'b0 ) | (bnz_op & rf_a != 16'b0))
     pc <= pc +{{8{imm[7]}},imm}+1 ;
   else
     pc <= pc+1;
end

endmodule
```