情報工学科 計算機構成 2010 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. 16bit RISC POCO で下の命令を順に実行した。

```
LDI r1, #0xff
ADDIU r1, #1
LD r3,(r1)
LD r2,(r3)
ST r2,(r1)
```

ここで、データメモリ中の値 (演習では dmem.dat) を 16 進数で以下のように設定した。ただし、本来はスペースでなく改行されているとする。

```
0003 0000 0001 0005 0002 0009 0028
```

a) 上記の機械語命令を示せ。またそれぞれの opcode フィールド、immediate フィールド (あれば) はどうなるかを示せ
b) r1、r2、r3 の値はどのように変化するかを示せ。また、最終的に、どの番地にどのような値が書き込まれるかを示せ。

2. 図に示した 16bit RISC POCO のデータパスについて答えよ。
(1) メモリ、レジスタファイルのアクセス時間をそれぞれ 10ns、8ns、ALU、マルチプレクサの遅延時間をそれぞれ 8ns、3ns とすると、動作周波数はいくつになるか？
(2) メモリから読み出したデータをレジスタに書き込むためには、どの制御線にどのようなレベルを与えれば良いか？
(3) メモリへ書き込みを行うためには、どの制御線にどのようなレベルを与えれば良いか？
3. 16bit RISC POCO について以下のプログラムをアセンブリ言語で記述せよ。機械語に変換する必要はない。また飛び先にはラベルを利用せよ。
(a) データメモリの 0 番地から 7 番地までに格納されている 8 個の 127 以下の正の数のうち最小値を求めるプログラムを記述せよ。
(b)r0 に与えられた番地から格納されている r1 中の個数分の 127 以下の正の数の中から最小値を求めて、r2 に答えを返すサブルーティンを記述せよ。
(c) スタックポインタが r6 であり既に値が設定されているとして、サブルーティン内で破壊するレジスタのみを退避して復帰するコードを (b) に付け加えよ。
4. 添付の POCO の Verilog コードは基本的命令のみのものである。このコードに以下の変更を行え。変更部分のみ示せ。
(a) rd,rs を比較して $rd \geq rs$ が成り立てば、(rd-rs がプラスになれば) 次の命令を飛ばしてさらに次の命令を実行するスキップ命令 SKIPR rd,rs を R 型で実装せよ。funct は空いているものの中から適当に選べ。
(b) LDLI2 rd, X 命令は、rd の下位に X をセットし、上位は元の rd の値を保持する命令である。この命令を実行できるように Verilog コードを改変せよ。opcode は空いているものの中から適当に選べ。

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
```

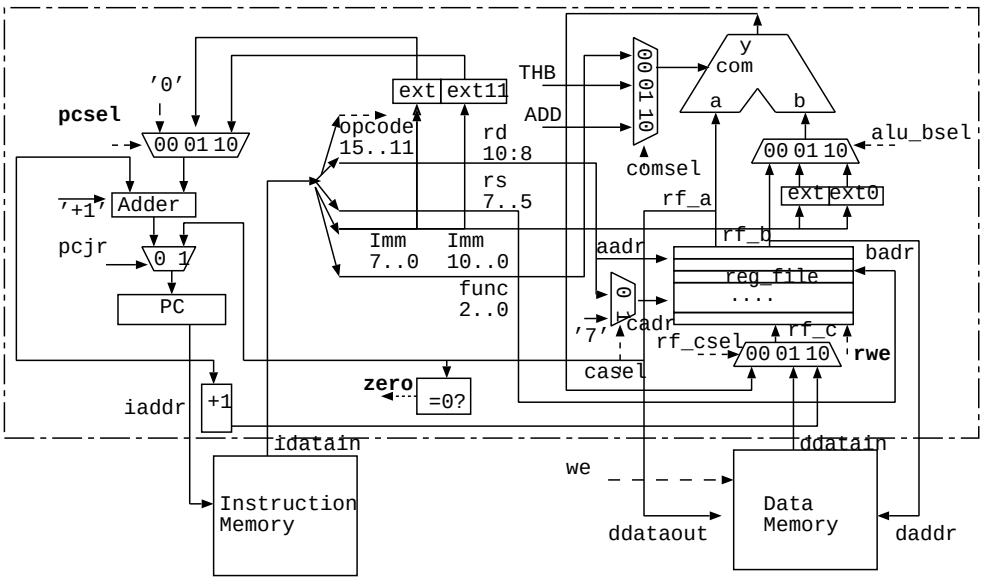

図 1: POCO のデータパス

```
‘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
```