情報工学科 計算機構成 2009 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。

```
LDIU r1, #0x02
LD r2,(r1)
LD r3,(r2)
ADDI r3, #0x80
ST r3,(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。

```
0005 0000 0001 0030 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、function(func) 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. 16bit RISC POCO で　a)LDI 命令、b)LDIU 命令、c)ADDIU 命令を実行する際に、各制御信号線をどのように設定すれば良いか？表に付け加えよ。

表 1: 各命令の制御信号

	pcsel	pcjr	comsel	alu_bsel	rf_csel	casel	rwe	we
ADDI	00	0	10	01	00	0	1	0

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) でデータメモリの 0 番地から 7 番地までに格納されている 8 個のデータの総和を求めて 8 番地に答えを格納す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 中の個数分のデータの総和を格納して、答えを r2 に格納して戻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) データメモリと命令メモリを統合して一つのメモリとして扱えるようにせよ。
(b) LDHI2 rd, X 命令は、rd の上位に X をセットし、下位は元の rd 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
```

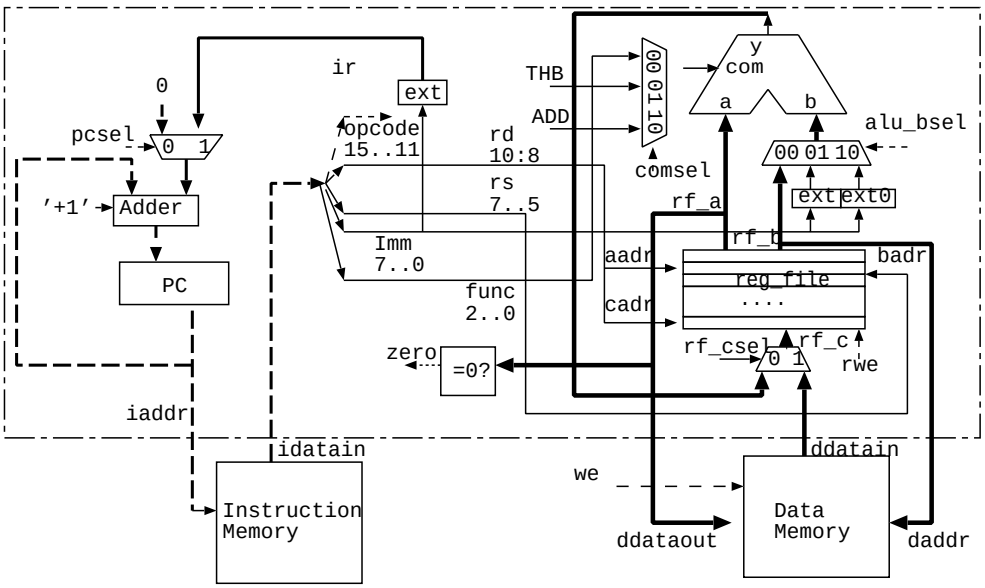

図 1: POCO のデータパス

```
‘include "def.h"
module poco(
input clk, rst_n,
input [‘DATA_W-1:0] idatain, ddatain,
output [‘DATA_W-1:0] iaddr, daddr, ddataout, output we_n);
reg [‘DATA_W-1:0] pc, ir;
reg [‘STAT_W-1:0] stat;
wire [‘DATA_W-1:0] rf_a, rf_b, rf_c, alu_b, alu_y;
wire [‘OPCODE_W-1:0] opcode, func;
wire [‘REG_W-1:0] rs, rd; wire [‘SEL_W-1:0] com; wire [‘IMM_W-1:0] imm;
wire pcsel, rwe;
wire st_op, bez_op, bnz_op, bmi_op, addi_op, ld_op, alu_op;
wire ldi_op, ldiu_op, ldhi_op, addiu_op;
assign ddataout = rf_a;
assign iaddr = pc;
assign daddr = rf_b;
assign {opcode, rd, rs, func} = ir;
assign imm = ir[‘IMM_W-1:0];
// Decorder
assign st_op = stat[‘EX] & (opcode == ‘OP_REG) & (func == ‘F_ST);
assign ld_op = stat[‘EX] & (opcode == ‘OP_REG) & (func == ‘F_LD);
assign alu_op = stat[‘EX] & (opcode == ‘OP_REG) & (func[4:3] == 2’b00);
assign ldi_op = stat[‘EX] & (opcode == ‘OP_LDI);
assign ldiu_op = stat[‘EX] & (opcode == ‘OP_LDIU);
assign addi_op = stat[‘EX] & (opcode == ‘OP_ADDI);
assign addiu_op = stat[‘EX] & (opcode == ‘OP_ADDIU);
assign ldhi_op = stat[‘EX] & (opcode == ‘OP_LDHI);
assign bez_op = stat[‘EX] & (opcode == ‘OP_BEZ);
assign bnz_op = stat[‘EX] & (opcode == ‘OP_BNZ);
assign bmi_op = stat[‘EX] & (opcode == ‘OP_BMI);
assign we_n = ~st_op;
assign alu_b = (addi_op | ldi_op) ? {{8{imm[7]}},imm} :
(addiu_op | ldiu_op) ? {8’b0,imm} :
(ldhi_op) ? {imm, 8’b0} : rf_b;
assign com = (addi_op | addiu_op ) ? ‘ALU_ADD:
(ldi_op | ldiu_op | ldhi_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 | ldhi_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));
assign pcsel = (bez_op & rf_a == 16’b0 ) | (bnz_op & rf_a != 16’b0) |
               (bmi_op & rf_a[15] == 1 ) ;
```

```
always @(posedge clk or negedge rst_n)  begin
   if(!rst_n) pc <= 0;
   else if(pcsel)
     pc <= pc +{{8{imm[7]}},imm} ;
   else if(stat[`IF])
     pc <= pc+1;  end
always @(posedge clk or negedge rst_n)  begin
   if(!rst_n) ir <= 0;
   else if(stat[`IF])
     ir <= idatain; end
always @(posedge clk or negedge rst_n)  begin
   if(!rst_n) stat <= `STAT_IF;
   else
    case (stat)
      `STAT_IF: stat <= `STAT_EX;
      `STAT_EX: stat <= `STAT_IF;
    endcase end
endmodule
```