

情報工学科 計算機構成 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 r0, #0x80
LD r1, (r0)
LD r2, (r1)
LD r3, (r2)
ADD r3, r2
```

ここで、データメモリ中の値 (演習では dmem.dat) を 0 番地から順に 16 進数で以下のように設定し、その他の番地はすべて 0 で埋めた。

```
0004 0000 0001 0005 0002 0009 0028
```

- 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
- r0, r1, r2, r3 の値はどのように変化するかを示せ。

2. 図に示した 16bit RISC POCO のデータパスについて答えよ。

- メモリ、レジスタファイルのアクセス時間をそれぞれ 2ns、1ns、ALU、マルチプレクサの遅延時間をそれぞれ 2.5ns、0.5ns、レジスタファイルの書き込みに対するセットアップ時間を 0.2ns とすると、動作周波数はいくつになるか？
- メモリのアクセス時間が 4ns の場合、動作周波数はいくつになるか？
- JAL 命令を実行するには、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。

- r0 中の数の 1 と 0 を反転させて r1 に答えを返すサブルーチンを記述せよ。
 - このサブルーチンを用いてメモリ中の 0 番地から並んだ 100 個のデータの 1 と 0 を反転させて元の番地に格納するプログラムを書け。
 - 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、サブルーチ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。このコードを変更し、比較命令 CMP rd, rs を付け加えよ。この命令は、 $rd > rs$ が成立すれば r7 を 1、そうでなければ 0 にする命令である。opcode は空いているものの中から適当に選べ。
5. module four は 8bit の入力 idata が 4 の倍数の時に hit=1、そうでなければ hit=0 を出力するハードウェアである。Verilog のコードを示せ。

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 XXXXXXXXXXXX
JAL X	r7 ← pc, pc ← pc + X	10101 XXXXXXXXXXXX
JR rd	pc ← rd	00000 ddd --- 01010

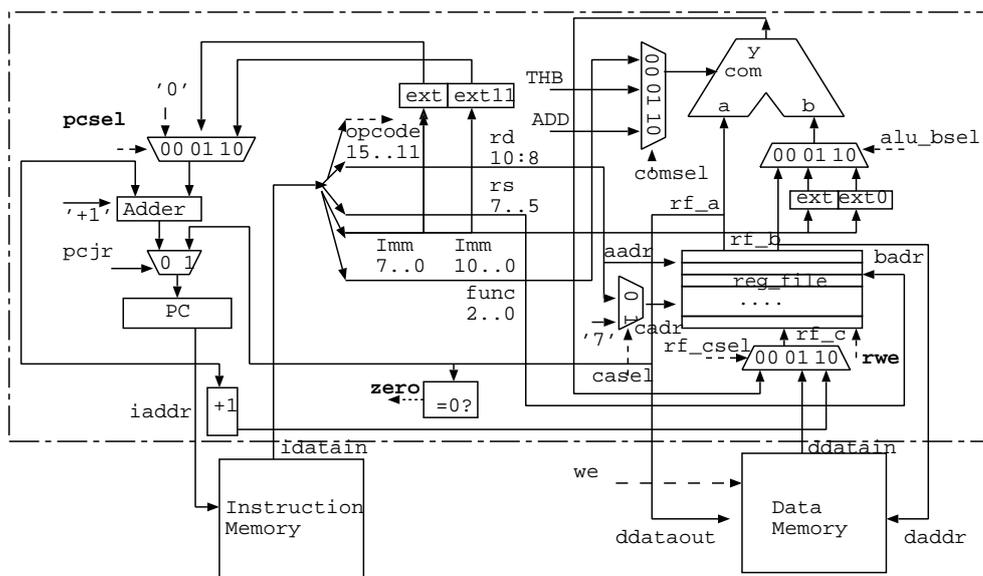


図 1: POCO のデータパス

```

`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];

// Decoder
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), .addr(rd), .b(rf_b), .baddr(rs),
.c(rf_c), .caddr(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

```