情報工学科 計算機構成 2011 年度期末試験解答例 (担当:天野 持ち込み何でも可)

正解はこれだけとは限りません。部分点はそれなりにあげています。

1. 16bit RISC POCO で下の命令を順に実行した。

```
LDI r0, #0x80   01000 000 10000000 Opcode 01000 Immediate 10000000
LD r1,(r0)     00000 001 000 01001 以下 Opcode は 00000
LD r2,(r1)     00000 010 001 01001
LD r3,(r2)     00000 011 010 01001
ADD r3,r2      00000 011 010 00110
```

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
```

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　（答：図中に示す各 1 点で 11 点）
b) r0, r1、r2, r3 の値はどのように変化するかを示せ。（各 2 点、r3 のみ 3 点で 9 点）

答：　r0 は ff80 になる。ここには 0 が入っているので、r1 は 0 になる。r2 は 0 番地の 4 が入り r3 は 4 番地の 2 が入る。これに r2 が加算されるので答えは 6 になる。

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 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？　（7 点）

答： ALU のバスは、2+1+0.5+2.5+0.5+0.2 = 6.7ns
データメモリを通るバスは、2+1+2+0.5+0.2 = 5.7ns
長い方をとって 1/6.7=149MHz (7 点)
セットアップ時間が入っていないと 1 点減点

(2) 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？

答： ALU のバスは、4+1+0.5+2.5+0.5+0.2 = 8.7ns
データメモリを通るバスは、4+1+4+0.5+0.2 = 9.7ns
長い方をとって 1/9.7=103MHz (7 点)
セットアップ時間が入っていないと 1 点減点

(3)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？(6 点)

答：pcjr=0 pcsel=10 rf_csel=10 casel=1 rwe =1 we=0 alu_bsel=X comsel=X X は don't care

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)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。(7 点)

答：

```
bitr  :LDI r1,0xff
       SUB r1,r0
       JR r7
```

(b) 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。(7 点)

答：

```
     LDIU r2,#0
     LDIU r3,#100
loop:LD r0,(r2)
     JAL bitr
     ST r1,(r2)
     ADDI r2,#1
     ADDI r3,#-1
     BNZ r3,loop
end: JMP end
```

(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。(6 点)

答：このサブルーチンならば特に退避する必要はない。

ただし、b をサブルーチンの形にしてこれに退避、復帰コードをつけた答案も正解とした。

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。(20 点)

答：

```
wire cmp_op;
wire [‘REG_W-1:0] cadr;
wire [‘DATA_W-1:0] cdata;

wire cmp_op = (opcode == ‘OP_REG) & (func == ‘F_CMP)
..
assign com = (addi_op | addiu_op ) ? ‘ALU_ADD:
                (ldi_op | ldiu_op ) ? ‘ALU_THB:
                cmp_op ? ‘ALU_SUB: func[‘SEL_W-1:0];

assign rwe = ld_op  | alu_op | ldi_op | ldiu_op | addi_op | addiu_op
               | cmp_op  ;
```

```
assign cadr = cmp_op ? 3'b111 : rd;

assign cdata= /|rf_a[14:0] & ~rf_a[15] ? 16'h1: 16'h0;

assign rf_c = cmp_op ? cdata : ld_op  ? ddatain : alu_y;

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

$rd > rs$ の条件は、15 ビット目が 0 で 14-0 ビット目のどこかに 1 があること。やや条件が難しかったかも。でもここではあまり減点していない。ちなみにこの命令はあまり良いアイディアではなく bmi, bpl の方がましである。

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。(20 点)

答：

```
module four (
  input [3:0] idata,
  output hit )

assign hit = ~(idata[1] | idata[0]);
endmodule
```

これも超簡単。hit をレジスタとしても良い。

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
```

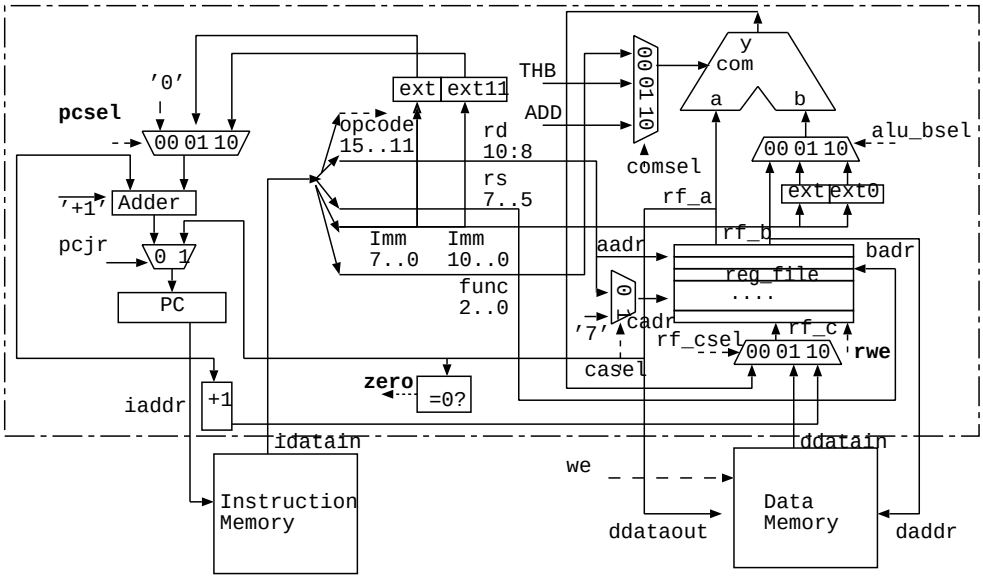

図 1: POCO のデータパス

```
‘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
```