情報工学科 計算機構成 2012 年 期末試験 (担当:天野 持ち込み何でも可)

ここに示す答はあくまで一例で、他にも正解はたくさんあります。

1. 16bit RISC POCO で 0 番地の内容から 1 番地の内容を引き算し 2 番地に結果を格納するプログラムをアセンブリ言語で記述せよ。
2. 問題 1 のプログラムを機械語に変換せよ。
答：LDIU は LDI でも良いですし、ADDI、ADDIU でも可能です。

```
LDIU r0,#0  01001_000_00000000
LD r1,(r0)  00000_001_000_01001
LDIU r0,#1  01001_000_00000001
LD r2,(r0)  00000_010_000_01001
SUB r1,r2   00000_001_010_00111
LDIU r0,#2  01001_000_00000010
ST r1,(r0)  00000_001_000_01000
```

3. 4bit の入力 month は 2 進数で「1 月〜12 月」までの月を表す。31 日まである「大の月」が入力された際に 1 を出力するモジュールを Verilog HDL で記述せよ。
答：もちろんマルチプレクサ構文でも OK です。大の月は 1 月、3 月、5 月、7 月、8 月、10 月、12 月です。これを知らない人が居ることは想定していませんでした。しかし大学生を相手にしているのだから、そこまで問題に書いてあげる必要はないでしょう。

```
module dainotuki
(input [3:0] month,
 output dai ) ;

dai = (month == 4'b0001) |
        (month == 4'b0011) |
        (month == 4'b0101) |
        (month == 4'b0111) |
        (month == 4'b1000) |
        (month == 4'b1010) |
        (month == 4'b1011) ;
endmodule
```

4. 16bit RISC POCO を目標周期 6ns で合成した結果、slack が 0.8ns 生じた。実際の最大動作周波数を求めよ。
答：1/(6-0.8)nsec = 192MHz
  6+0.8 にした人は、0 点です。これはしょーがないですよね。
5. 下のサブルーチンの先頭で退避したレジスタを復帰するコードを書け。

```
subr:
   ADDI r6,#-1
   ST r2,(r6)
   ADDI r6,#-1
   ST r5,(r6)
...
```

答: 問題ではバカなミスをしてすいません。最後の JR は別に書かなくてもいいです。

```
   LD r5,(r6)
   ADDI r5,#1
   LD r2,(r6)
   ADDI r6,#1
   JR r7
```

6. 0 番地から 8 つの正の整数が並んでいる。このうち 10 より大きい（10 は含まない）ものの個数を調べるプログラムを書け。結果は r6 に入れよ。
答: BPL は 0 でも飛ぶので、10 はカウントされないです。

```
      LDIU r0,#0
      LDIU r2,#8
      LDIU r6,#0
loop: LD r1,(r0)
      ADDI r1,#10
      BPL r6,skip
      ADDI r6,#1
skip: ADDI r0,#1
      ADDI r2,#-1
      BNZ r2,loop
```

7. 添付の POCO のデータパスにおいて、クリティカルパスに成り得る経路を二つ示せ。
答：いろいろな表記法がありますが、意味が分かるものは全て OK としました。

1. PC-命令メモリ-idatain-badr-ALU 入力-ALU 出力-レジスタファイル入力
2. PC-命令メモリ-idatain-badr-データメモリ-レジスタファイル入力

8. 添付の POCO の Verilog 記述を読み、rwe が 1 になる命令は何かを示せ。
答：LD、ALU 命令 (ADD, SUB, AND, OR など)、LDI, LDIU, ADDI, ADDIU
9. 64K ワードのアドレス空間に対して 2K ワードのキャッシュを設ける。ブロックアドレスを 4 ワードとした時、2way 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。

答：ダイレクトマップならば 2K/4=512 個ラインが入る、2way にするとセットは半分になるので 256 セットとなる。このためインデックスは 8bit、キーは 6bit となる。このディレクトリを 2 個分設ければ良い。
10. ライトバックキャッシュのキャッシュディレクトリ (タグメモリ) に Dirty bit を設けることの利点について簡単に説明せよ。

答：ライトバックキャッシュは、書き込みを行われていないブロックを主記憶に書き戻す必要はない。Dirty bit を設けて書き込みが行われたかどうかを記録することで、不必要な書き戻しを防ぎ、これにより性能低下を防ぐことができる。
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
```

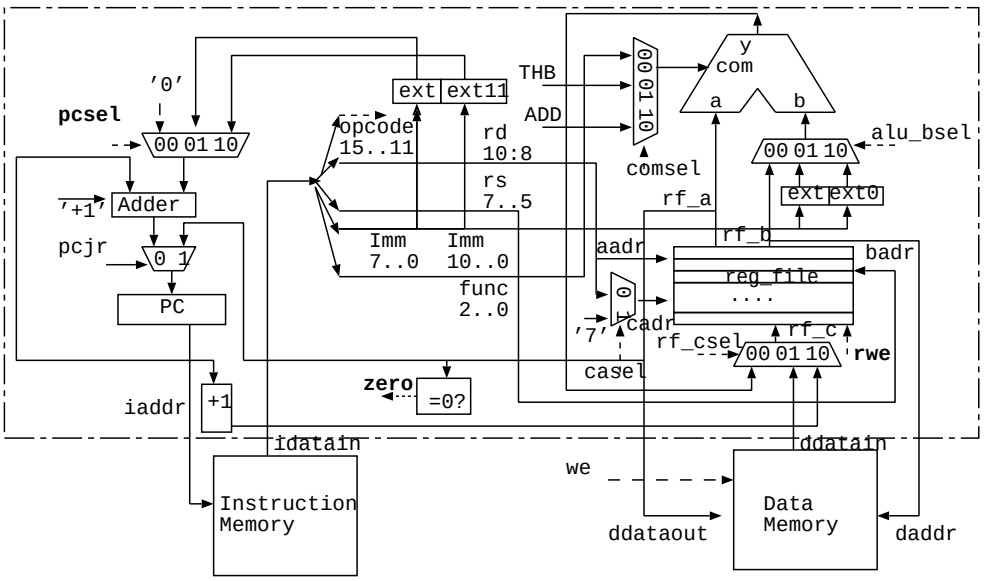

図 1: POCO のデータパス

```
‘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
```