情報工学科 計算機構成 2014 年度 期末試験回答例

ここでの答はあくまで一例です。

1. 0 番地に A, 1 番地に B が入っている。$(A-B)OR(A>>1)$ を計算し、答えを 2 番地に入れる POCO のプログラムをアセンブリ言語で記述せよ。

```
LDI r0,#0
LD r1,(r0)
MV r3,r1
ADDI r0,#1
LD r2,(r0)
SUB r1,r2
SR r3,r3
OR r1,r3
ADDI r0,#1
ST r1,(r0)
```

解説：一度 A の内容をどこかに保存しておく必要があります。ここでは r3 を使ってます。もちろん 2 回取ってきてもいいです。

2. 問題 1 のプログラムの最初の 4 行を機械語に変換せよ。

```
01000_000_00000000
00000_001_000_01000
00000_011_001_00001
01100_000_00000001
```

3. A, B それぞれ 2bit の入力のうち A が B より大きい場合に出力 Y を 1 にするモジュールを Verilog HDL で記述せよ。

```
module agtb (
input [1:0] a,b,
output y);
assign y = (a>b) ? 1'b1: 1'b0;
endmodule
```

解説：assign 文は単に以下でも OK です。

```
assign y = (a>b) ;
```

4. 0 番地から 99 番地までの 100 個の数を、100 番地から 199 番地に移動するプログラムを POCO のアセンブラで書け。

```
     LDIU r0, #99
     LDIU r1, #199
loop:LD r2,(r0)
     ST r2,(r1)
```

```
     ADDI r1,#-1
     ADDI r0,#-1
     BPL r0,loop
end: JMP end
```

解説：ここでは r0 をループ制御に使っているが、別にカウンタを使ってもいいです。
5. サブルーチン div は、r0/r1 を実行し、答えを r2、余りを r3 に返す。このサブルーチンを利用し、0 番地に格納されている数が 3 の倍数ならば、1 番地に 1 を、そうでなければ 0 を書くプログラムを POCO のアセンブラで書け。

```
     LDI r4,#0
     LD r0,(r4)
     LDI r1,#3
     JAL div
     LDI r5,#1
     BNZ r3,store
     LDI r4,#1
store:ST r4,(r5)
end: JMP end
```

解説：余りが 0 の時に 1 を返す必要があるのでちょっとごちゃごちゃします。ここでは r4 が 0 であることを利用しています。
6. POCO を、目標周期を 10ns にして論理合成した所、slack が-0.4ns になった。そこで目標周期を 12ns にした所 slack が 0.4ns になった。両者の動作周波数を求めよ。
1/(10+0.4) = 96.2MHz
1/(12-0.4) = 86.2MHz
7. 問題 6 の前者と後者のうちどちらが面積（ゲート数）が大きくなると考えられるか？またそれはなぜかを説明せよ。
前者が大きくなる。速度を上げようとして合成すると、合成系は高速加算器など高速だがゲート数の多い回路を使うためゲート数が増えてしまう。
8. 添付の POCO の Verilog 記述でレジスタファイルの書き込みを制御する信号の名称を書け。この信号はどのような命令で 1 にするかを説明せよ。
rwe。この記述では LD 命令、ALU 命令 (MV,ADD など 8 種類)、LDI 命令、LDIU 命令、ADDI 命令、ADDIU 命令
9. 256Kword の主記憶に対して 32Kword のキャッシュを設ける。ブロックサイズを 64word とした時の (1) ダイレクトマップキャッシュのディレクトリ構成、(2) 2-way セットアソシアティブキャッシュのディレクトリ構成を述べよ。
32K は 2 の 15 乗で、64word のキャッシュブロックが 2 の 9 乗個、すなわち 512 個入る。index は 9bit である。256Kword は 2 の 18 乗なので tag は 18-6-9=3bit となる。
ダイレクトマップ 3bit 幅 512 の Tag memory X1
2way 4bit 幅 256 の Tag memory X2
10. 最近のプロセッサのキャッシュが階層構成になっている理由を簡単に説明せよ。
CPU の動作周波数とメモリの性能差が大きくなり、単一のキャッシュではその差を吸収できないため階層を増やし、ミスペナルティを小くする。
全 10 点

A) POCO の命令コード

```
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
```

B) POCO の Verilog 記述

```
‘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, 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 rwe, 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
```