情報工学科 計算機構成 2010 年 期末試験解答例

ここに示したのは一例で、リーゾナブルな答えには相応の点が与えられています。

1. 16bit RISC POCO で下の命令を順に実行した。

```
LDI r1, #0xff
ADDIU r1, #1
LD r3,(r1)
LD r2,(r3)
ST r2,(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。

```
0003 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 (機械語 10 点、フィールド 5 点)

01000 001 11111111 opcode 01000 immediate 11111111
01100 001 00000001 opcode 01100 immediate 00000001
00000 011 001 01001 opcode 00000
00000 010 011 01001 opcode 00000
00000 010 001 01000 opcode 00000

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。(各 2 点で 10 点)

r1 ffff $\rightarrow$ 0
r2 5
r3 3
0 番地に 5 が書き込まれる。

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 10ns、8ns、ALU、マルチプレクサの遅延時間をそれぞれ 8ns、3ns とすると、動作周波数はいくつになるか？(11 点)
この問題は大変出来が悪かった。間違いの原因が分かるものには部分点をあげているが、ただ数字を書かれるとあげようがない。

ALU 演算の遅延は、10(命令メモリ)+8(レジスタファイル)+3(b 入力マルチプレクサ)+8(ALU)+3(rf_c 入力マルチプレクサ) = 32nsec

データメモリアクセスの遅延は、10(命令メモリ)+8(レジスタファイル)+10(データメモリ)+3(rf_c 入力マルチプレクサ) = 31nsec

よって長い方を考えて、1/32 = 31.25MHz

(2) メモリから読み出したデータをレジスタに書き込むためには、どの制御線にどのようなレベルを与えれば良いか？(7 点)

we=0, rwe=1, rf_csel=01, casel=0

(3) メモリへ書き込みを行うためには、どの制御線にどのようなレベルを与えれば良いか (7 点)？

we=1, rwe=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) 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 127 以下の正の数のうち最小値を求めるプログラムを記述せよ。(10 点)

```
      LDI r0,#7
      LD r1,(r0)
loop: ADDI r0,#-1
      LD r2,(r0)
      SUB r2,r1
      BPL r2, skip
      ADD r1,r2
skip: BNZ r0,loop
```

(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 127 以下の正の数の中から最小値を求めて、r2 に答えを返すサブルーティンを記述せよ。(10 点)

ここでは最大数である１２７をまず r2 に入れておき、これと比較する。すべての数はこれ以下なので、このプログラムはうまく行くはず。

```
min:  LDIU r2,#127
loop: LD r3,(r0)
      SUB r3,r2
      BPL r3, skip
      ADD r2,r3
skip: ADDI r0,#1
      ADDI r1,#-1
      BNZ r1,loop
      JR r7
```

(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。(5 点)

r3,r1,r0 を退避する。

```
min:  ADDI r6,#-1
      ST r3,(r6)
```

```
       ADDI r6,#-1
       ST r1,(r6)
       ADDI r6,#-1
       ST r0,(r6)

      LDIU r2,#127
loop: LD r3,(r0)
       SUB r3,r2
       BPL r3, skip
       ADD r2,r3
skip: ADDI r0,#1
       ADDI r1,#-1
       BNZ r1,loop

       LD r0,(r6)
       ADDI r6,#1
       LD r1,(r6)
       ADDI r6,#1
       LD r3,(r6)
       ADDI r6,#1
       JR r7
```

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) rd,rs を比較して $rd \geq rs$ が成り立てば、(rd-rs がプラスになれば) 次の命令を飛ばしてさらに次の命令を実行するスキップ命令 SKIPR rd,rs を R 型で実装せよ。funct は空いているものの中から適当に選べ。(13 点)

```
assign skipr_op = (opcode == `OP_REG) & (func == `F_SKIPR);
assign com = (addi_op | addiu_op ) ? `ALU_ADD:
                (ldi_op | ldiu_op ) ? `ALU_THB:
                (skipr_op) ? `ALU_SUB: func[`SEL_W-1:0];
...

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 if (skipr_op & ~alu_y[15]) pc <= pc +2;
   else
     pc <= pc+1;
end
```

この問題はいきなり出てきたら難問だが、演習で一度やったので解けてもいいと思う。
(b) LDLI2 rd, X 命令は、rd の下位に X をセットし、上位は元の rd の値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。(12 点)

```
assign ldli2_op = (opcode == `OP_LDLI2);
...
assign alu_b = (addi_op | ldi_op) ? {{8{imm[7]}},imm} :
                (addiu_op | ldiu_op) ? {8'b0,imm} :
                (ldli2_op) ? {rf_a[15:8],imm} : rf_b;
assign com = (addi_op | addiu_op ) ? `ALU_ADD:
                (ldi_op | ldiu_op | ldii2_op ) ? `ALU_THB: func[`SEL_W-1:0];

...
assign rwe = ld_op  | alu_op | ldi_op | ldiu_op | addi_op | addiu_op | ldli2_op  ;
```

これも似たような問題を演習でやったので解けると思ったのだが大変出来が悪かった。