情報工学科 計算機構成 2009 年度期末試験解答例

答えはあくまで一例です。一部の誤りについての部分点は付いてます。

1. 16bit RISC POCO で下の命令を順に実行した。

```
LDIU r1, #0x02
LD r2,(r1)
LD r3,(r2)
ADDI r3, #0x80
ST r3,(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。

```
0005 0000 0001 0030 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、function(func) フィールド (あれば) はどうなるかを示せ

答:

```
01001 001 00000010 : opcode 01001
00000 010 001 01001 : opcode 00000 func 01001
00000 011 010 01001 : opcode 00000 func 01001
01100 011 10000000 : opcode 01100
00000 011 001 01000 : opcode 00000 func: 01000
```

各命令についてコード化 3 点、opcode/func 2 点で 25 点

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。

答: まず最初の LDLI 命令で r1 が 2 となる。次の LD 命令で 2 番地の 0001 が r1 にセットされる。次の LD 命令で 1 番地の 0 が r3 にセットされる。ここで、0x80 が加算されるが符号拡張が行われるので、0xff80 が r3 にセットされ、これが r1 で示された 2 番地に格納されることになる。答えは、

r1: 2, r2: 1 r3: 0 → 0xff80, 2 番地に 0xff80 が書き込まれる　各 2 点で 8 点

2. 16bit RISC POCO で　a)LDI 命令、b)LDIU 命令、c)ADDIU 命令を実行する際に、各制御信号線をどのように設定すれば良いか？表に付け加えよ。

答: 問題となるのは ALU の comsel と alu_bsel。前者は ADDI, ADDIU では加算をするために 10、LDI, LDIU では、B 入力をスルーするために 01 とする。後者は、LDI では符号拡張するので 01、LDIU, ADDIU では上位 8 ビットを 0 で詰めるため 10 となる。各 6 点で計 18 点

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。

表 1: 各命令の制御信号

	pcsel	pcjr	comsel	alu_bsel	rf_csel	casel	rwe	we
ADDI	00	0	10	01	00	0	1	0
LDI	00	0	01	01	00	0	1	0
LDIU	00	0	01	10	00	0	1	0
ADDIU	00	0	10	10	00	0	1	0

(a) でデータメモリの 0 番地から 7 番地までに格納されている 8 個のデータの総和を求めて 8 番地に答えを格納するプログラムを記述せよ。

答:

```
       LDLI r0,#0   // address
       LDLI r1,#8   // counter
       LDLI r2,#0   // result
loop:  LD r3,(r0)   // read data
       ADD r2,r3    // add
       ADDIU r0,#1  // address increment
       ADDI r1, #-1  // counter decrement
       BNZ r1,loop   // if counter != 0 goto loop
end:   BEZ r1,end    // dynamic stop
```

上記はあくまで一例。動作するものは皆正解である。ダイナミックストップは入れなくてもいい。10 点

(b)r0 に与えられた番地から格納されている r1 中の個数分のデータの総和を格納して、答えを r2 に格納して戻すサブルーティンを記述せよ。

答:

```
       LDLI r2,#0   // result
loop:  LD r3,(r0)   // read data
       ADD r2,r3    // add
       ADDIU r0,#1  // address increment
       ADDI r1, #-1  // counter decrement
       BNZ r1,loop   // if counter != 0 goto loop
       JR r7
```

10 点。サブルーティンなので、JR r7 がないと 2 点減点。

(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を退避して復帰するコードを (b) に付け加えよ。

答:

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

```
        ADDI r6,#-1
        ST r1,(r6)
        ADDI r6,#-1
        ST r3,(r6)
        LDLI r2,#0   // result
loop:   LD r3,(r0)   // read data
        ADD r2,r3    // add
        ADDIU r0,#1  // address increment
        ADDI r1, #-1  // counter decrement
        BNZ r1,loop   // if counter != 0 goto loop
        LD r3,(r6)
        ADDI r6,#1
        LD r1,(r6)
        ADDI r6,#1
        LD r0,(r6)
        ADDI r6,#1
        JR r7
```

5 点。r2 は結果が入るレジスタなので格納してはならない。r2 を格納したら 1 点減点。

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) データメモリと命令メモリを統合して一つのメモリとして扱えるようにせよ。

答 : まずは状態マシンを作る。

```
reg [`STAT_W-1:0] stat;
...
always @(posedge clk or negedge rst_n)
begin
   if(!rst_n) stat <= `STAT_IF;
   else
    case (stat)
      `STAT_IF: stat <= `STAT_EX;
      `STAT_EX: stat <= `STAT_IF;
    endcase
end
```

次に ir を設けて、命令フェッチができるようにする。(計 13 点)

```
reg [`DATA_W-1:0] ir;
assign addr = stat[`IF] ? pc: rf_b;
assign {opcode, rd, rs, func} = ir;
assign imm = ir[`IMM_W-1:0];
...
always @(posedge clk or negedge rst_n)
begin
```

```
   if(!rst_n) ir <= 0;
   else if(stat[‘IF])
     ir <= datain;
end
```

主要な変更は上記で OK。その他もしも ALU で pc を+1 にする場合はもっとコードが必要になる。この改造は授業ではやったが、もちろん今回の題意からするとやらなくても良い。

(b) LDHI2 rd, X 命令は、rd の上位に X をセットし、下位は元の rd 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。

答: レジスタの値を imm と組み合わせて、b 入力に与える方法が標準的。9 点

```
assign ldhi2_op = stat[‘EX] & (opcode == ‘OP_LDHI2);
..
assign alu_b = stat[‘IF] ? 16’h1:
                (addi_op | ldi_op | branches) ? {{8{imm[7]}},imm} :
                jumps ? {{5{ir[10]}},ir[10:0]} :
                (addiu_op | ldiu_op) ? {8’b0,imm} :
                (ldhi_op) ? {imm, 8’b0} :
               (ldhi2_op) ? {imm, rf_a[7:0]}: rf_b;
...
assign com = (stat[‘IF] | branches| jumps | addi_op | addiu_op ) ? ‘ALU_ADD:
                (ldi_op | ldiu_op | ldhi_op | ldhi2_op) ? ‘ALU_THB:
                                                        func[‘SEL_W-1:0];
```