1. 略
2. 127+63=189 をアセンブルしてみた

```
000001_00000_00001_0000000001111111_
000001_00000_00010_0000000000111111_
000000_00010_00001_00011_00000000000_
```

3. 3.1 名称 bgt0_sub
   アセンブリ言語による表現 sub rt, rt, rs + blt r0, rt, dpl
   動作 rt<-rt-rs をして rt>0 なら pc ← (pc)+4+dpl
   書式 R 型
   コード 36
   3.2 変更箇所周辺のみ記述する

```
elsif($op eq "ble") { p_b(6, 35); p_r2b($f2, $f3);
 p_b(16, $labels{$f4} - $i - 1); print("\n"); }

elsif($op eq "bgt0_sub") { p_b(6, 36); p_r2b($f2, $f3);
 p_b(16, $labels{$f4} - $i - 1); print("\n"); }

elsif($op eq "j"  ) { p_b(6, 40); p_b(26, $labels{$f2}); print("\n"); }
```

   3.3 ここも変更箇所周辺のみ記述する

```
function [4:0] opr_gen;
input [5:0] op;
input [4:0] operation;
case (op)
      // R 型はこっち
6'd0: opr_gen = operation;
      // I 型で ALu を用いるのは以下
6'd1: opr_gen = 5'd0;
6'd4: opr_gen = 5'd8;
6'd5: opr_gen = 5'd9;
6'd6: opr_gen = 5'd10;
      // 変更箇所 (bgt0_sub)
      6'd36: opr_gen = 5'd2;
default: opr_gen = 5'h1f;
endcase
endfunction
```

```
・
・
・
function [31:0] calc;
input [5:0] op;
input [31:0] alu_result, dpl_imm, dm_r_data, pc;
case (op)
      // 変更箇所 (bgt0_sub)
6'd0, 6'd1, 6'd4, 6'd5, 6'd6, 6'd36: calc = alu_result;
6'd3: calc = dpl_imm << 16;
・
・
・

function [31:0] npc;
input [5:0] op;
input [31:0] reg1, reg2, branch, nonbranch, addr;
case (op)
6'd32: npc = (reg1 == reg2) ? branch : nonbranch;
6'd33: npc = (reg1 != reg2) ? branch : nonbranch;
6'd34: npc = (reg1 < reg2)  ? branch : nonbranch;
6'd35: npc = (reg1 <= reg2) ? branch : nonbranch;
      // 変更箇所 (bgt0_sub)
      6'd36:  npc = (reg1 > reg2)  ? branch : nonbranch;
6'd40, 6'd41: npc = addr;
・
・
・
function [4:0] wreg;
input [5:0] op;
input [4:0] rs, rt, rd;
case (op)
6'd0: wreg = rd;
6'd1, 6'd3, 6'd4, 6'd5, 6'd6, 6'd16, 6'd18, 6'd20: wreg = rt;
      // 変更箇所 (bgt0_sub)
      6'd36: wreg = rs;
```

3.4
```
addi r1, r0, 10
addi r2, r0, 1
addi r3, r0, 0
```

```
loop_start : add r3, r2, r3
addi r2, r2, 1
ble r2, r1, loop_start
j, loop_end
loop_end : addi, r3, r3, 0
```
を翻訳して
```
00001_00000_00001_0000000000001010_
000001_00000_00010_0000000000000001_
000001_00000_00011_0000000000000000_
000000_00010_00011_00011_00000000000_
000001_00010_00010_0000000000000001_
100011_00010_00001_1111111111111101_
101000_00000000000000000000000111_
000001_00011_00011_0000000000000000_
```

3.5
```
addi r1, r0, 10
addi r2, r0, 1
addi r3, r0, 0
loop_start : add r3, r1, r3
bgt0_sub,r1,r2, loop_start
j, loop_end
loop_end : addi, r3, r3, 0
```
を翻訳して
```
000001_00000_00001_0000000000001010_
000001_00000_00010_0000000000000001_
000001_00000_00011_0000000000000000_
000000_00001_00011_00011_00000000000_
100100_00001_00010_1111111111111110_
101000_00000000000000000000000110_
000001_00011_00011_0000000000000000_
```

3.6 bgt0_sub を導入したことで 1 ループで 1 クロック性能向上するので、性能向上は N