What you think about this code snippet? It seems that Z88DK compiler faile with "unsigned >" comparison but "unisgned <" works okay. May it be bug of compiler?

```
      case OPSBA_BLTU:
        if ((REGU_TYPE) *_rs1_ < (REGU_TYPE) *_rs2_)
{ core.pc += imm << LSHIFT; }
        break;

      case OPSBB_BLEU: //pseudo
        if ((REGU_TYPE) *_rs1_ <= (REGU_TYPE)
*_rs2_) { core.pc += imm << LSHIFT; }
        break;

      case OPSBC_BGEU:
        if ((REGU_TYPE) *_rs1_ >= (REGU_TYPE)
*_rs2_) { core.pc += imm << LSHIFT; }
        break;

      case OPSBD_BGTU: //pseudo
        //HACK for Z88DK Z80 compiler - "unsigned
>" FAILS !!!
        //HACK - we use reversed registers and "<"
here !!!
        if ((REGU_TYPE) *_rs2_ < (REGU_TYPE) *_rs1_)
{ core.pc += imm << LSHIFT; }
        break;
```

The code snippet you've provided seems to be implementing a form of branch logic based on unsigned comparisons, likely within the context of a virtual machine or emulator, as it appears to be manipulating a program counter (`core.pc`).

The fact that you've had to resort to a "hack" for the `OPSBD_BGTU` case, where you reverse the registers and use the `<` operator instead of `>` due to an apparent issue with the Z88DK compiler handling of `unsigned >` comparison, is interesting.

I cannot say definitively whether it's a compiler bug without further details. Such an issue would be quite surprising given that this is a very fundamental part of the C language which should be handled correctly by any mature compiler.

However, it's also not impossible. Compiler bugs do exist, even in mature and widely used compilers. You might want to make sure you're using the latest version of the compiler, and if the issue persists, consider reaching out to the Z88DK community or the compiler maintainers with a minimal, complete, and verifiable example of the issue.

Also, just to make sure, ensure that there aren't any macro or type definition conflicts which could be causing this issue. The code you've shown us uses macros and pointers extensively, and sometimes, issues can arise from the interactions between these elements. For example, `*_rs1_` is a pointer dereference, and if `_rs1_` doesn't point to a valid location or there's something wrong with the casting to `REGU_TYPE`, that could be causing the issue.

In conclusion, this could be a compiler bug, but it's also possible that it could be an issue with the code itself, the compiler version you're using, or some interaction between various parts of your code. You'll need to do further debugging to isolate and identify the source of the issue.