1 AkariDoc

This is TeX documentation of testing framework of porting Intel intrinsic function to RISC-V or ARM or IBM POWER.This is one of solution of Intel architecutre dependency problem.

2 Problem

Intel x86-64 is CISC,this has long history and carryes a lot of heritage.CISC like Intel x86-64 is converting CISC instruction to micro op of RISC inside microprosessor.So we need circuit for it and it makes die size large and has a lot of overhead.On the other hand,RISC is relatively new so it carries less heritage and we can use relatively new technology.But we have many source code run only on Intel x86-64.So we need to port Intel x86-64 code to RISC.One source of CPU architecture dependency problem is Intel Intrinsic function.This project is intended to port Intel x86-64 intrinsic function to RISC-V or ARM or IBM POWER or other architecture.

3 Solution

Connect Intel x86-64 and RISC by network. At first local area network.And input Intel intrinsic function and corespondance of that of ported to RISC same value. And make sure return value is same.

4 CPU architecture dependency problem!!

We are using Intel x86-64 PC or server.Most cloud server use Intel x86-64 CPU.But Intel x86-64 is not power efficient and it's performance is not good.But Why we are using x86-64? Because we have a lot of software assets runs on x86-64 in this world.

Most software assets is written by C language.We can understand assembler is CPU architecture dependent but by C language is too? C language can be compiled both RISC-V and ARM and Intel.The source of CPU architecture dependency is Intel Intrinsic function.This is "Intel" intrinsic function so it only runs on Intel x86-64, not RISC-V or ARM or IBM POWER.

But How to port Intel intrinsic function to other platform?The porting process need few ingenuity and great effort.Make wrapping structure like below.

Listing 1 hoge

```
extern __inline __m128d __attribute__((__gnu_inline__, __always_inline__,
    __artificial__))
_mm_add_pd (__m128d __A, __m128d __B)
{
  return (__m128d) ((__v2df)__A + (__v2df)__B);
```

```
    }
```