Computer Architecture: Privilege Instruction System **Fall 2024**

Homework 3 — September 23

Lecturer: Hu Weiwu *Completed by: Zhang Jiawei*

3.1

在处理完异常之后，需要执行异常返回指令以恢复现场。异常返回指令原子地完成恢复权限等级、恢复中断使能状态、跳转至异常返回目标等多个操作。

LoongArch 中的 ERTN 指令就是如此，它会将 CSR. PRMD 的 PPLV 和 PIE 域分别回填至 CSR. CRMD 的 PLV 和 IE 域, 从而使得 CPU 的权限等级和全局中断响应状态恢复到异常发生时的状态, 同时该指令还会将 CSR. ERA 中的值作为目标地址跳转过去。这样就完成了恢复现场的工作。

3.2

1. 存放 EPTR 的位置不同：在 LoongArch 中，EPTR 存放在 CSR. ERA 中；在 x86 中，EPTR 以 CS 和 EIP 组合的形式存放在栈中。

2. 确定异常来源的方式不同：在 LoongArch 中，通过将不同的异常进行编号，以“入口页号与页内偏移进行按位逻辑或”的方式计算出相应异常处理程序的入口地址进行处理；在 x86 中，通过查询 IDT 表中的描述符，找到相应异常处理程序的入口地址进行处理。

3.3

精确异常是指在发生任何异常时，被异常打断的指令之前的所有指令都执行完，而该指令之后的所有指令都像没执行一样。异常处理程序可忽略因处理器流水线带来的异常发生位置问题。异常处理结束后将返回 EPTR 所在地址，重新执行被异常打断的指令。也就是说，EPTR 所指向的指令就是异常发生时的指令。

非精确异常是指在发生异常时，无法保证被异常打断的指令之前的所有指令都已经执行完毕，或者无法保证被异常打断的指令之后的所有指令都没有执行，异常处理程序返回的 EPTR 所指向的指令可能不是异常发生时的指令。

在某些需要多个时钟周期才能完成的指令（如浮点运算指令）中，如果在执行过程中发生异常，那么这个异常就是非精确异常。因为浮点运算通常需要多个时钟周期才能完成，而在此期间，处理器可能会继续执行其他指令，这样就无法保证异常发生时的指令是否已经执行完毕。

3.4

每个数组占 $65536 \times 8 = 512KB$，即 128 页。

第一次循环，由于所有页表中都没有数据且 V 项为 0，所以会产生 2*128/2 = 128 次重填异常、128 次 store 页无效异常和 128 次 load 页无效异常。

第二、三次循环，仅会发生重填异常，共 2*128 = 256 次。

所以，总共会产生 640 次异常。

3.5

```
struct {
   boolean VPPN[`VPPN];
```

```
    boolean PS[`PS];
    boolean G;
    boolean ASID[`ASID];
    boolean E;
} TLB[`TLB_SIZE];
struct {
    boolean PPN[`PPN];
    boolean RPLV[`RPLV];
    boolean PLV[`PLV];
    boolean MAT[`MAT];
    boolean NX;
    boolean NR;
    boolean D;
    boolean V;
} PTE[`PTE_SIZE];

int hit = 0;
int i = 0;
while(!hit){
    if (TLB[i].E == 1 && TLB[i].VPPN == vppn && (TLB[i].G == 1 || TLB[i].ASID ==
        asid)){
        hit = 1;
        break;
    }
}
if (hit){
    findPTE(TLB[i].VPPN);
    if (PTE[addr].V == 0){
        if (load)
            return PIL;
        else if (store)
            return PLS;
    }
    else if (PTE[addr].RPLV == 0 && CSR.CRMD.PLV > PTE[addr].PLV || PTE[addr].RPLV
        == 1 && CSR.CRMD.PLV != PTE[addr].PLV)
        return PPI;
    else if (load && PTE[addr].NR == 1)
        return PNR;
    else if (store && PTE[addr].D == 0)
        return PME;
    else if (instFetch)
        return PNX;
    else{
```

```
        paddr = {PTE[addr].PPN, offset};
        type = PTE[addr].MAT;
        return POK;
    }
}
```