VM HW4 B10902078

Hackmd link: link

From the QEMU docs, we know that QEMU handles the guest memory access translations and permission checks during the TLB lookup. When there is a TLB miss, the helper function `store_helper()` will call `tlb_fill()` in `accel/tcg/cputlb.c`.

```
ok = cc->tcg_ops->tlb_fill(cpu, addr, size,
                           access_type, mmu_idx, false, retaddr);
```

In `tlb_fill()`, it will call the architecture-specific `tlb_fill()`. Since the target architecture we're emulating is arm64, so we can get the specific `tlb_fill()` for arm64, which is specified in `target/arm/cpu.c` like below:

```
#else
    .tlb_fill = arm_cpu_tlb_fill,
    .cpu_exec_interrupt = arm_cpu_exec_interrupt,
```

Next, we can check the `arm_cpu_tlb_fill()` function located in `target/arm/tlb_helper.c`. Inside, we can see it call

```
ret = get_phys_addr(&cpu->env, address, access_type,
                        core_to_arm_mmu_idx(&cpu->env, mmu_idx),
                        &phys_addr, &attrs, &prot, &page_size,
                        &fi, &cacheattrs);
```

Since ARM uses the LPAE (Large Physical Address Extension) format, we can get the corresponding `get_phys_addr_lpae()` function.

My modification:

```
static bool get_phys_addr_lpae(...) {
    ...

    fault_type = ARMFault_Permission;
    if (!(*prot & (1 << access_type))) {

        // add bypass logic
```

```
        if ((address & ~(page_size - 1)) == (0x4005e4 & ~(page_size - 1))) {
                *prot |= PAGE_WRITE;
        }
        else{
                goto do_fault;
        }
    }
    ...
}
```

Since the page containing the victim code is marked as read-only, so I add a bypass logic in `get_phys_addr_lpae()`. Specifically, since permissions are handled at page granularity in MMU, so if the querying virtual address is located at the same page as our target address `0x4005e4`, I will set the page to include write permissions.