CS502 PROJECT2

Chengjiao Yang   12/11/2013

Architectural And Policy Overview

This project focus on the disk and the memory mechanism. we are required to practice the memory and disk write and read of the Z502.c, based on that, we have to design the page fault handle, page fetch and replacement algorithm. Since the physical memory is too less to deal with the large numbers of data, Make sure the data walks well among virtual memory, physical memory, disk is important part and well-designed in this project. Beyond that, the muti-processes test are applied. For more detail, please see the context below.

What Is Included

1. Communicate with hardware
Fault_handler, Interrupt_handler
2. Memory manage
frametable, MEMORY, GetFreeFrame(), IsFreeFrameExist() for memory control
3. Disk
WriteToDisk(), ReadFromDisk() for disk write and read
4. Process synchronization
Integrate lock to prevent the conflict
5. Print & Debug
dospprint(), Memory_Print() to print the schedule and the memory management information

High Level Design

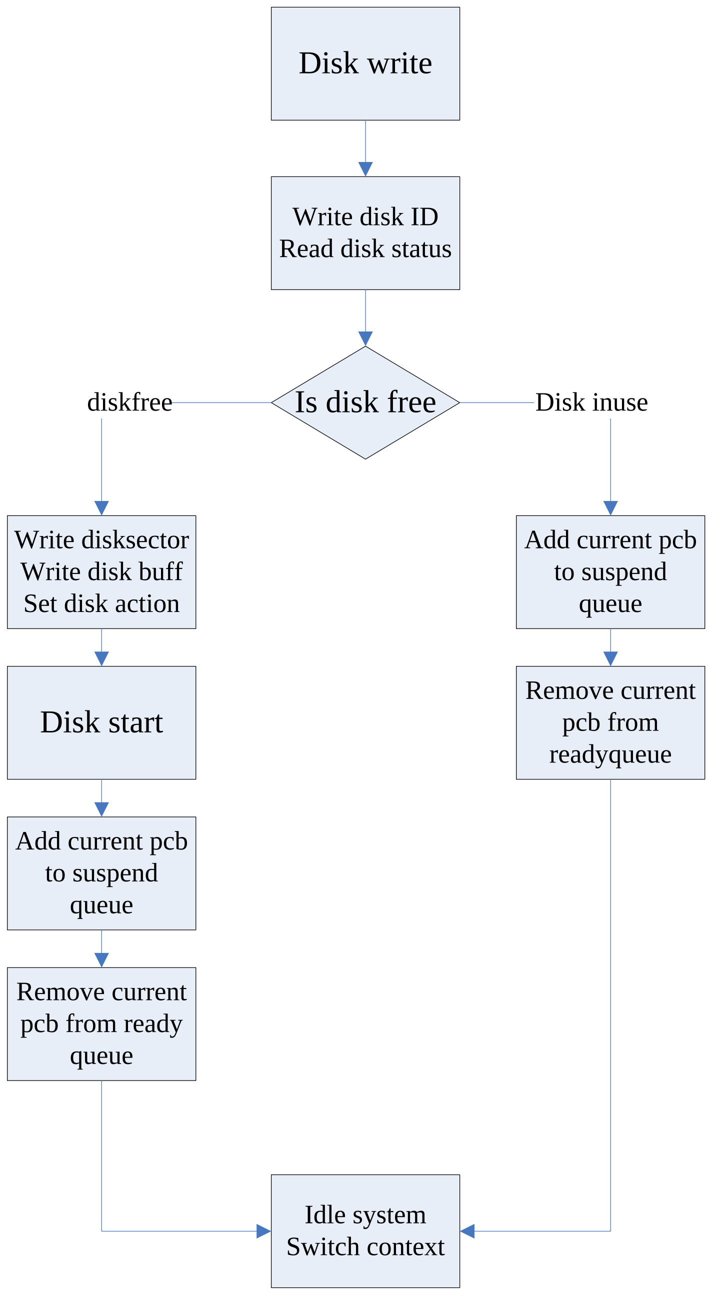

figure 1

Justification: WriteToDisk(), ReadFromDisk()

This diagram show how the disk mechanism works, when the user request a disk write, system call to OS, the SVC get the request and dispatch the request to WriteToDisk() routine. Because the hardware cost time to handle the disk write and read, so we first have to judge if the disk is free or in_use. If disk free, init the disk parameter, then add PCB to suspend status to Reserve time for hardware response, then simply idle system until the next disk interrupt occur. ReadFromDisk() is the routine handle the read request, its almost the same as WriteToDisk(), the only difference is setting the disk action to 0.

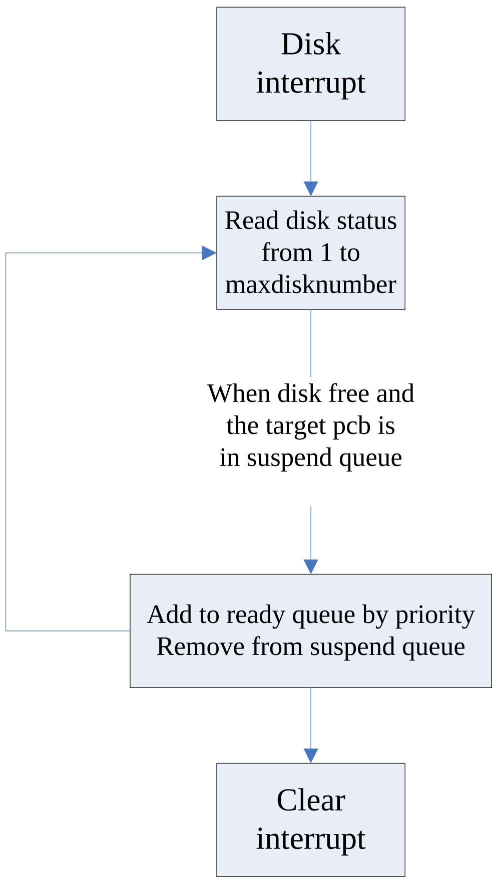
Disk interrupt
Read disk status from 1 to maxdisknumber
When disk free and the target pcb is in suspend queue
Add to ready queue by priority Remove from suspend queue
Clear interrupt

figure 2

Justification: Disk interrupt

This diagram show what the disk interrupt does. The disk interrupt types range from 5 to 5 plus MAX_NUMBER_OF_DISKS. Since the hardware only allow one interrupt at the same time, some interrupt may lose during the system running. So every time we get disk interrupt request, we check the status for all the disks, if disk is free and the target PCB is in suspend queue, it means the hardware has finished the write/read request of the target PCB, then we could move all those PCB from suspend queue to readyqueue. Of course, make sure to clear interrupt after all.

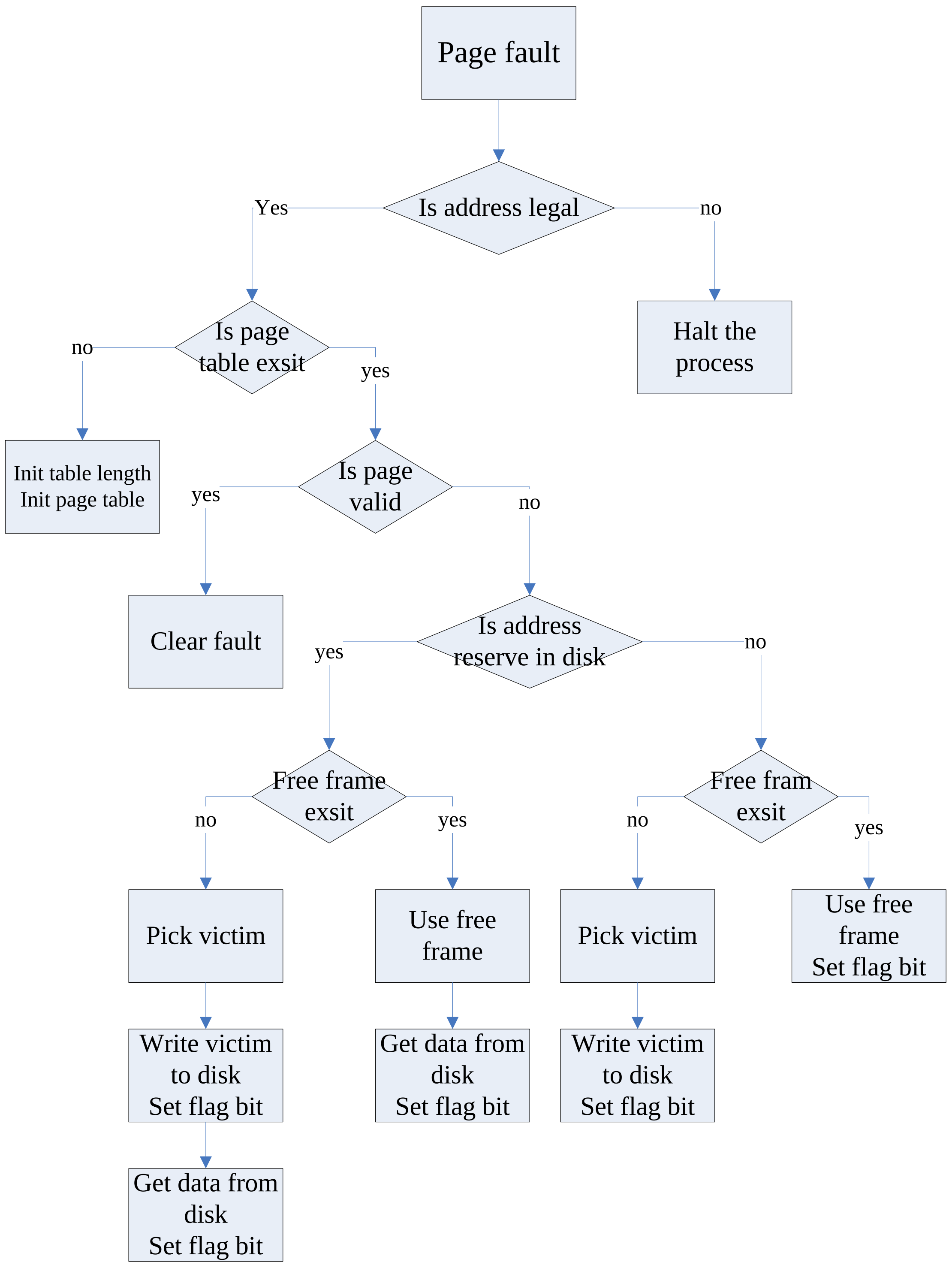

figure 3

Justification: fault handler, memory fetch and replacement, page and frame table

This diagram show the core part of this project. When anything is wrong during handling the memory(device_id is invalid_memory) request, it goes to the fault_handler.

First we need to find the reasons of the fault. If its because the request address is large than virtual memory or the request address is smaller than 0, its an illegal address, we simply halt system. If no page table was found, we init the Z502_PAGE_TBL_ADDR and Z502_PAGE_TBL_LENGTH for current PCB.

Secondly, according to the virtual memory structure(figure 4), we check the valid bit

Valid Bit	Modified Bit	Referenced Bit	Reserved (1 Bit)	Physical Page Number (12 Bits)

figure 4

of the target address, if it's valid, it should not be here^^. Otherwise, the reserved bit tells us if the target is in disk, if the target reserve in disk, we should copy this address to physical memory, if the target is not reserved in disk, it means the hardware just need a memory space, we can just allocate a physical memory address. No matter its in disk or not, we have to check physical frame, here I use the frametable to manage the physical frame, use IsFreeFrameExist() and GetFreeFrame() routine to judge the frame status and get free frame. If there is free frame, we can happy to use this frame, if not, we need a algorithm to pick a most less possible used frame as a victim, write the victim to disk and allocate memory space. During all those operation, every time we change the status of the virtual memory, we need to change the flag bit.

Unique About The Project

1.To distinguish the time interrupt and disk interrupt, I use "DISK_INT" to mention its a disk interrupt, "TIME_INT" to show its a timer interrupt.

2.Since there are so much disks but less memory and processes, I simply match one process with one disk, pid0 = disk1, pid1 = disk2 like these. Every disk has 1600 sectors, for this reason, I didn't use the shadow page table, I handle the VPN the same as the sector number. Maybe set a shadow page table to store the address of disk sectors to match the virtual memory is a good way to save space. However, we have so many disk space, in addition, disk is cheaper, so why not do it.

3.About test2g, I could run it, but the test still can't work well. It always have a correct result at first several loops, then the first error go out, then the result getting worse and worse. I think its because the error snowball, the reason to form this problem should be the lock mechanism, because it works well in single process. I tried to set lock in fault_handler in several ways, unfortunately, the problem is not fixed.

Additional Features

In this project, I use the Approximate LRU algorithm for picking the victim memory. Since the Z502 virtual memory has already support a reference bit to us, it seems simple to use the algorithm like LRU.

Every time the system request a physical memory, we need to find a victim, originally we use FIFO method, I use global variable "currentvictim" to save the current FIFO number. The LRU algorithm is based on FIFO, when we pick the victim, we also check the reference bit of the virtual memory, if it's 1, then set this bit to 0, give this victim another chance. If it is 0, choose this one as victim this time, write this physical memory to disk. However, the algorithm seems not runs very well at test2f, the test result show there are 1046 write and 987 read, but there are 1110 faults. When I simply use virtual address mod PHYS_MEM_PGS to pick victim, it only have 1057 faults.

Anomalies Or Bugs

1.Schedulerprint can only show 4 number of time, that’s no convince.

2.System still doesn't output the same result among VS, GCC(Mingw), Linux. I use VS as IDE, every time I could run project well in VS, but it doesn't work well in GCC. After adding lock mechanism, it was getting better, however, some tests still hard to pass through, especially the test2e and test2f.

3. When I run project in VS, it appears "idle for ever, no event will cause an interrupt" error sometimes. There is a magic command "freopen("filename.txt", "w", stdout);" to solve this problem. If I add this command to code to output the result, this error is hardly happened. After checking the occur time of the interrupt, I found the Z502DiskStart can start even the interrupt haven't end yet. But after trying some code to solve this, the problem still the same. Then I found another magic command: "printf("");" When I add this magic command before CALL(Z502Idle()), something become running well. See example below.

```
while(IsEmpty(readyqueue)){ //while nothing in readyqueue, do idle
    printf("");
    CALL(Z502Idle());
}
```