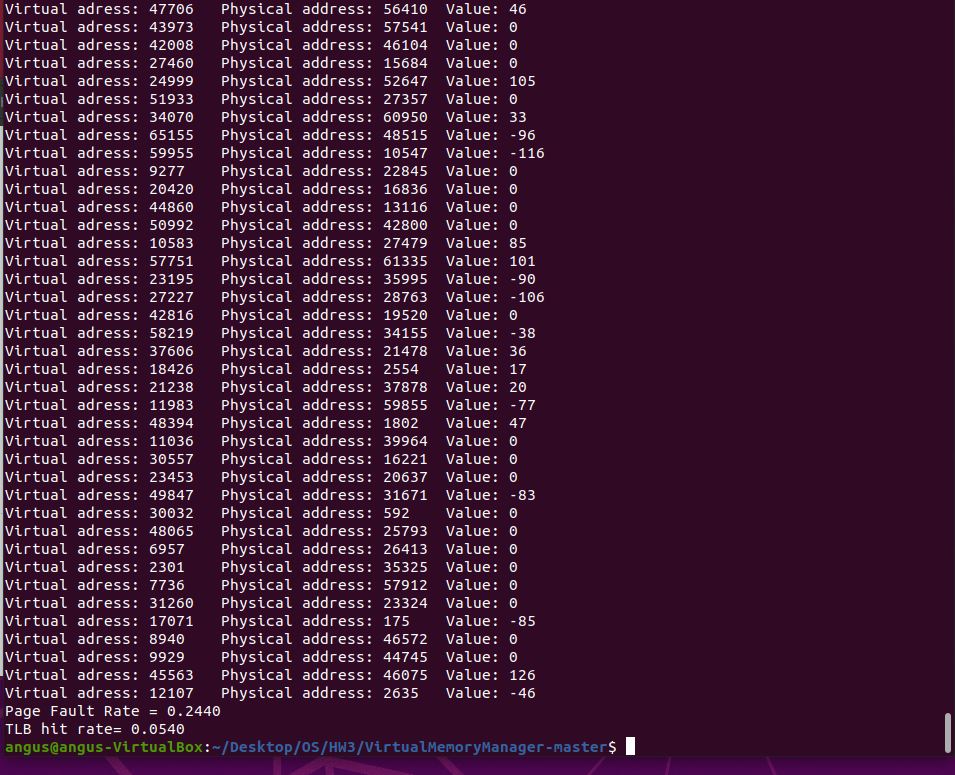
Steps：

1. Make
2. ./a.out addresses.txt



Source code：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

const int PAGE\_TABLE\_SIZE = 256;

const int BUFFER\_SIZE = 256;

const int PHYS\_MEM\_SIZE = 256;

const int TLB\_SIZE = 16;

struct TLB {

unsigned char TLBpage[16];

unsigned char TLBframe[16];

int ind;

};

int readFromDisk (int pageNum, char \*PM, int\* OF){

char buffer[BUFFER\_SIZE];

memset(buffer, 0, sizeof(buffer));

FILE \*BS;

BS = fopen("BACKING\_STORE.bin", "rb");

if (BS == NULL){

printf("File failed to open\n");

exit(0);

}

// printf("offset in fseek: %d\n", pageNum\*PHYS\_MEM\_SIZE);

if (fseek(BS, pageNum \* PHYS\_MEM\_SIZE, SEEK\_SET)!=0)

printf("error in fseek\n");

if (fread(buffer, sizeof(char), PHYS\_MEM\_SIZE, BS)==0)

printf("error in fread\n");

int i = 0;

for(i; i < PHYS\_MEM\_SIZE; i++){

\*((PM+(\*OF)\*PHYS\_MEM\_SIZE)+i) = buffer[i];

/\* printf("buffer[%d]=%d\n",i+pageNum\*256, buffer[i]);\*/

}

(\*OF)++;

return (\*OF)-1;

// printf("&d\n ", \*((PM+i\*n)+j));

// i = current row row number, n = elements per row, j = colum num

}

int findPage(int logicalAddr, char\* PT, struct TLB \*tlb, char\* PM, int\* OF, int\* pageFaults, int\* TLBhits){

unsigned char mask = 0xFF;

unsigned char offset;

unsigned char pageNum;

bool TLBhit = false;

int frame = 0;

int value;

int newFrame = 0;

printf("Virtual adress: %d\t", logicalAddr);

pageNum = (logicalAddr >> 8) & mask;

// printf("%X\t", pageNum);

offset = logicalAddr & mask;

// printf("%X\t", offset);

//Check if in TLB

int i = 0;

for (i; i < TLB\_SIZE; i++){

if(tlb->TLBpage[i] == pageNum){

frame = tlb->TLBframe[i];

TLBhit = true;

(\*TLBhits)++;

// printf("TLBhit\t\t");

}

}

//Check if in PageTable

if (TLBhit == false){

if (PT[pageNum] != -1){

// printf("Pagehit\t\t");

}

//if not in either read from disk

else{

// printf("pageFault\t");

newFrame = readFromDisk(pageNum, PM, OF);

PT[pageNum] = newFrame;

(\*pageFaults)++;

// tlb->TLBpage[tlb->ind] = pageNum;

// tlb->TLBframe[tlb->ind] = newFrame;

// tlb->ind = (tlb->ind + 1)%TLB\_SIZE;

}

frame = PT[pageNum];

tlb->TLBpage[tlb->ind] = pageNum;

tlb->TLBframe[tlb->ind] = PT[pageNum];

tlb->ind = (tlb->ind + 1)%TLB\_SIZE;

}

int index = ((unsigned char)frame\*PHYS\_MEM\_SIZE)+offset;

value = \*(PM+index);

printf("Physical address: %d\t Value: %d\n",index, value);

return 0;

}

int main (int argc, char\* argv[]){

int val;

FILE \*fd;

int openFrame = 0;

int pageFaults = 0;

int TLBhits = 0;

int inputCount = 0;

float pageFaultRate;

float TLBHitRate;

unsigned char PageTable[PAGE\_TABLE\_SIZE];

memset(PageTable, -1, sizeof(PageTable));

struct TLB tlb;

memset(tlb.TLBpage, -1, sizeof(tlb.TLBpage));

memset(tlb.TLBframe, -1, sizeof(tlb.TLBframe));

tlb.ind = 0;

char PhyMem[PHYS\_MEM\_SIZE][PHYS\_MEM\_SIZE];

if (argc < 2){

printf("Not enough arguments\nProgram Exiting\n");

exit(0);

}

fd = fopen(argv[1], "r");

if (fd == NULL){

printf("File failed to open\n");

exit(0);

}

//printf("Value\tPageNum\tOffset\n ");

while (fscanf(fd, "%d", &val)==1){

// printf("%d\t", val);

findPage(val, PageTable, &tlb, (char\*)PhyMem, &openFrame, &pageFaults, &TLBhits);

inputCount++;

}

// readFromDisk(0, (char\*)PhyMem, &openFrame);

/\* int i = 0;

for (i; i < PHYS\_MEM\_SIZE; i++)

printf("PhyMem[%d]=%d\n",i, PhyMem[0][i]);

\*/

pageFaultRate = (float)pageFaults / (float)inputCount;

TLBHitRate = (float)TLBhits / (float)inputCount;

printf("Page Fault Rate = %.4f\nTLB hit rate= %.4f\n",pageFaultRate, TLBHitRate);

close(fd);

return 0;

}