CSL301
12340220

Assignment: TLB and Page Fault Measurement in XV6

Part 1 – LRU

Step 1: Add `page_faults` field in `proc` struct

File: `proc.h`

```
struct frameinfo {
 uint va ; // Virtual address of page
 pte_t* pte ; // Page table entry
 int ref ; // Reference ( optional for CLOCK )
 uint last_used ; // Time of last access ( for LRU )
};
```

```
struct proc {
 ...
 struct frameinfo frames[16]; // Track resident pages ( up to 16)
 int framecount ; // Number of frames used
};
```

Step 2: Initialize `page_faults` in `allocproc()`

File: `vm.c`

```
extern uint ticks;
```

Change `allocuvm` function

```
int allocuvm(pde_t *pgdir, uint oldsz, uint newsz) {
 char *mem;
 uint a;

 if (newsz >= KERNBASE) return 0;
 if (newsz < oldsz) return oldsz;
```

```
 a = PGROUNDUP(oldsz);

 for (; a < newsz; a += PGSIZE) {
   mem = kalloc();
   if (mem == 0) {
     cprintf("allocuvm out of memory\n");
     deallocuvm(pgdir, newsz, oldsz);
     return 0;
   }
   memset(mem, 0, PGSIZE);
   if (mappages(pgdir, (char*)a, PGSIZE, V2P(mem), PTE_W|PTE_U) < 0) {
     cprintf("allocuvm out of memory (2)\n");
     deallocuvm(pgdir, newsz, oldsz);
     kfree(mem);
     return 0;
   }

   struct proc *curproc = myproc();
   if (curproc) {
     if (curproc->framecount < 16) {
       curproc->frames[curproc->framecount].va = a;
       curproc->frames[curproc->framecount].pte =
walkpgdir(pgdir,(void*)a,0);
       curproc->frames[curproc->framecount].last_used = ticks;
       curproc->framecount++;
     } else {
       // choose least-recently-used victim
       int victim = 0;
       for (int i = 1; i < curproc->framecount; i++)

         if (curproc->frames[i].last_used <
             curproc->frames[victim].last_used) victim = i;

       pte_t *vpte = curproc->frames[victim].pte
       uint pa = PTE_ADDR(*vpte);
       kfree(P2V(pa));
       *vpte = 0;

       curproc->frames[victim].va = a;
       curproc->frames[victim].pte = walkpgdir(pgdir,(void*)a,0);
       curproc->frames[victim].last_used = ticks;
     }
   }
 } return newsz; }
```

Add `update_lru_access` function

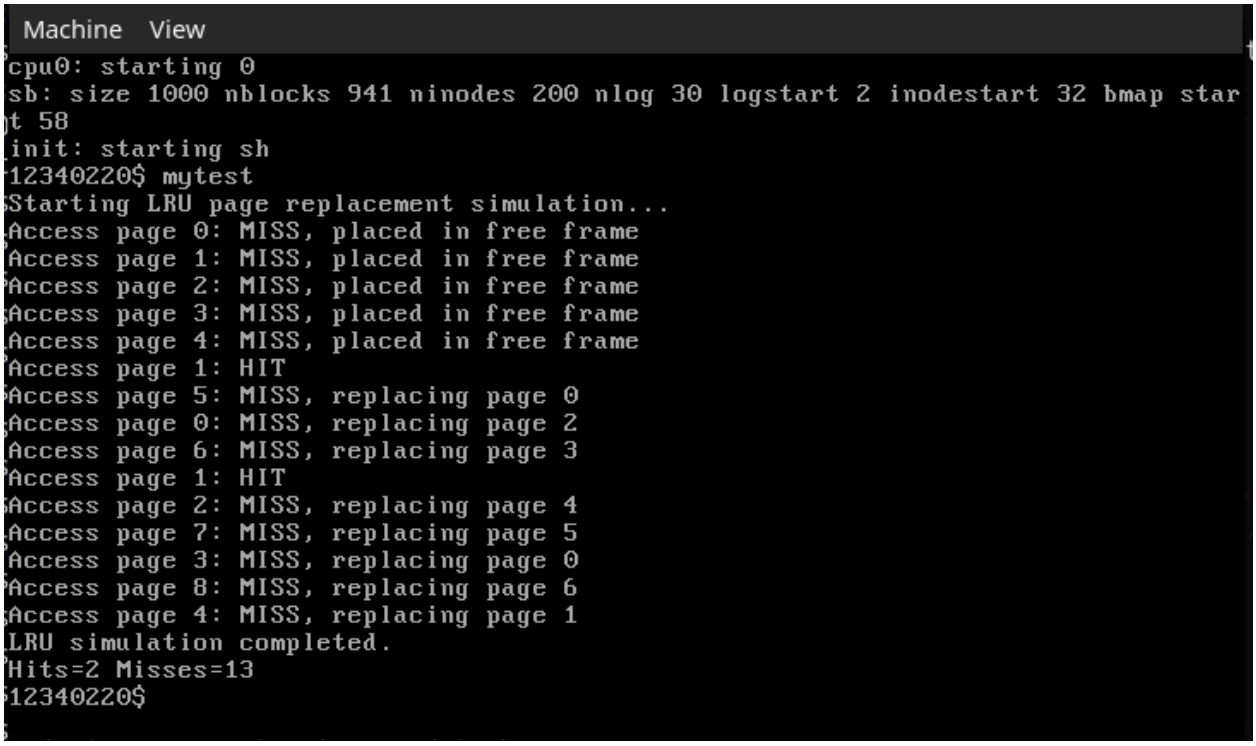
```
void update_lru_access (struct proc *p , uint va ) {
 for ( int i = 0; i < p->framecount ; i ++) {
   if (p->frames [ i ]. va == va ) {
     p->frames [ i ]. last_used = ticks ; // update on access
     break ;
   }
 }
}
```

Step 3: Update `mytest` function

```
#include "types.h"
#include "user.h"

#define NUM_PAGES 20       // Number of pages to allocate
#define PAGE_SIZE 4096     // 4 KB page size

int main(int argc, char *argv[])
{
   char *pages[NUM_PAGES];

   printf(1, "Starting FIFO page replacement test...\n");

   // Allocate multiple pages
   for(int i = 0; i < NUM_PAGES; i++){
       pages[i] = sbrk(PAGE_SIZE);
       if(pages[i] == (char*)-1){
           printf(1, "Allocation failed at page %d\n", i);
           break;
       }

       // Touch the page to actually allocate physical memory
       pages[i][0] = (char)i;

       printf(1, "Allocated page %d at virtual address 0x%x\n", i,
pages[i]);
   }

   printf(1, "FIFO test completed.\n");
```

```
    exit();
}
```

Observations



Part 2 – FIFO

Step 1: Add FIFO Fields in `proc.h`

File: `proc.h`

```
struct proc {
 ...
 // FIFO page replacement tracking
 uint pa_queue[MAX_PAGES];     // Physical addresses of allocated pages
 int queue_start;              // Index of oldest page
 int queue_end;                // Index for next page
 int queue_count;              // Number of pages in queue
};
```

Step 2: Update `vm.c` to include FIFO functions

```
// ----------------- FIFO PAGE QUEUE FUNCTIONS -----------------
#define MAX_FIFO_PAGES 64

struct fifo_page {
   void *va;
};

struct fifo_queue {
   struct fifo_page pages[MAX_FIFO_PAGES];
   int head;
   int tail;
   int count;
} fifo;

void fifo_init(void)
{
   fifo.head = 0;
   fifo.tail = 0;
   fifo.count = 0;
}

int add_page_to_fifo(void *va)
{
   if(fifo.count >= MAX_FIFO_PAGES)
       return -1; // FIFO full
   fifo.pages[fifo.tail].va = va;
   fifo.tail = (fifo.tail + 1) % MAX_FIFO_PAGES;
   fifo.count++;
   return 0;
}

void *remove_page_from_fifo(void)
{
   void *va;
   if(fifo.count <= 0)
       return 0;
   va = fifo.pages[fifo.head].va;
   fifo.head = (fifo.head + 1) % MAX_FIFO_PAGES;
   fifo.count--;
   return va;
}
```

Step 3: Update `mytest` function

```
#include "types.h"
#include "user.h"

#define MAX_PAGES 5
#define TOTAL_ACCESSES 15

int main(void) {
 int lru[MAX_PAGES], last_used[MAX_PAGES];
 int i, j, page, hit = 0, miss = 0, time = 0;

 for (i = 0; i < MAX_PAGES; i++) {
   lru[i] = -1;
   last_used[i] = -1;
 }

 printf(1, "Starting LRU page replacement simulation...\n");
 int accesses[TOTAL_ACCESSES] = {0,1,2,3,4,1,5,0,6,1,2,7,3,8,4};

 for (i = 0; i < TOTAL_ACCESSES; i++) {
   page = accesses[i];
   time++;
   int found = 0;

   for (j = 0; j < MAX_PAGES; j++)
     if (lru[j] == page) {
       found = 1;
       last_used[j] = time;
       break;
     }

   if (found) {
     hit++;
     printf(1, "Access page %d: HIT\n", page);
   } else {
     miss++;
     int free_index = -1;
     for (j = 0; j < MAX_PAGES; j++)
       if (lru[j] == -1) { free_index = j; break; }

     if (free_index != -1) {
       lru[free_index] = page;
       last_used[free_index] = time;
       printf(1, "Access page %d: MISS, placed in free frame\n", page);
     } else {
       int lru_idx = 0, min = last_used[0];
```

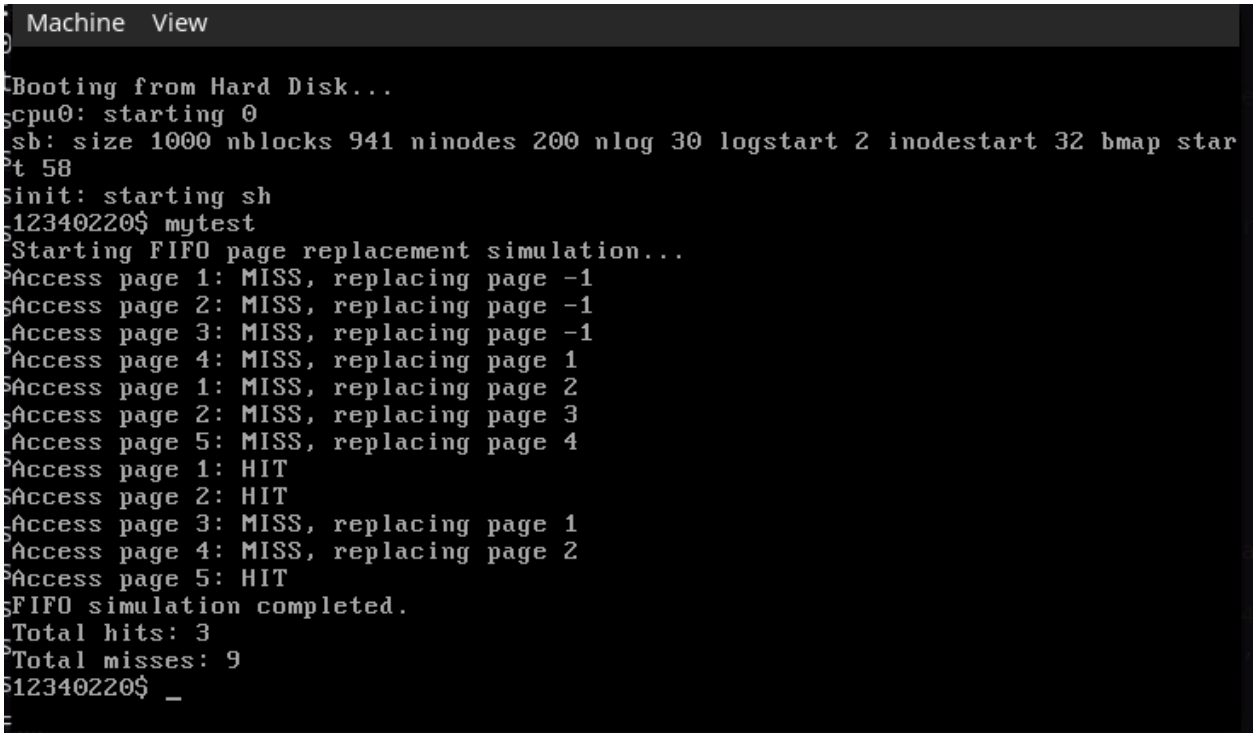
```
      for (j = 1; j < MAX_PAGES; j++)
        if (last_used[j] < min) { min = last_used[j]; lru_idx = j; }
      printf(1, "Access page %d: MISS, replacing page %d\n", page,
lru[lru_idx]);
      lru[lru_idx] = page;
      last_used[lru_idx] = time;
    }
  }
 }
 printf(1, "LRU simulation completed.\nHits=%d Misses=%d\n", hit, miss);
 exit();
}
```

Observations

For this following particular example
int accesses[TOTAL_ACCESSES] = {1,2,3,4,1,2,5,1,2,3,4,5};

For MAX_PAGES = 3

3 hits and 9 misses



For MAX_PAGES = 4

2 hits and 10 misses -> ANOMALY PRESENT HERE

```
Machine  View

Booting from Hard Disk...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
init: starting sh
12340220$ mytest
Starting FIFO page replacement simulation...
Access page 1: MISS, replacing page -1
Access page 2: MISS, replacing page -1
Access page 3: MISS, replacing page -1
Access page 4: MISS, replacing page -1
Access page 1: HIT
Access page 2: HIT
Access page 5: MISS, replacing page 1
Access page 1: MISS, replacing page 2
Access page 2: MISS, replacing page 3
Access page 3: MISS, replacing page 4
Access page 4: MISS, replacing page 5
Access page 5: MISS, replacing page 1
FIFO simulation completed.
Total hits: 2
Total misses: 10
12340220$
```

For MAX_PAGES = 5

7 hits and 5 misses -> no anomaly

```
Machine  View

Booting from Hard Disk...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
init: starting sh
12340220$ mytest
Starting FIFO page replacement simulation...
Access page 1: MISS, replacing page -1
Access page 2: MISS, replacing page -1
Access page 3: MISS, replacing page -1
Access page 4: MISS, replacing page -1
Access page 1: HIT
Access page 2: HIT
Access page 5: MISS, replacing page -1
Access page 1: HIT
Access page 2: HIT
Access page 3: HIT
Access page 4: HIT
Access page 5: HIT
FIFO simulation completed.
Total hits: 7
Total misses: 5
12340220$ _
```