Changes made for xv6-public-LRU:

(1) In proc.h, added the struct frameinfo:

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

Inside struct proc added:

```
  struct frameinfo frames[16];  // Track resident pages (max 16)
  int framecount;               // Number of frames used
```

(2) In vm.c, added at the top:

```
extern uint ticks;   // global tick counter from trap.c
```

Also added the following 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;
      break;
    }
  }
}
```

Also changed the 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;
    }

    // ---------------- LRU Update ----------------
    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 {
        // Select victim using LRU
        int victim_index = 0;
        for (int i = 1; i < curproc->framecount; i++) {
          if (curproc->frames[i].last_used < curproc->frames[victim_index].last_used) {
            victim_index = i;
          }
        }
        // Free victim
        pte_t *vpte = curproc->frames[victim_index].pte;
        uint vpa = PTE_ADDR(*vpte);
        kfree(P2V(vpa));
        *vpte = 0;

        // Replace with new page
        curproc->frames[victim_index].va = a;
        curproc->frames[victim_index].pte = walkpgdir(pgdir, (void*)a, 0);
        curproc->frames[victim_index].last_used = ticks;
      }
    }
  }
  return newsz;
}
```

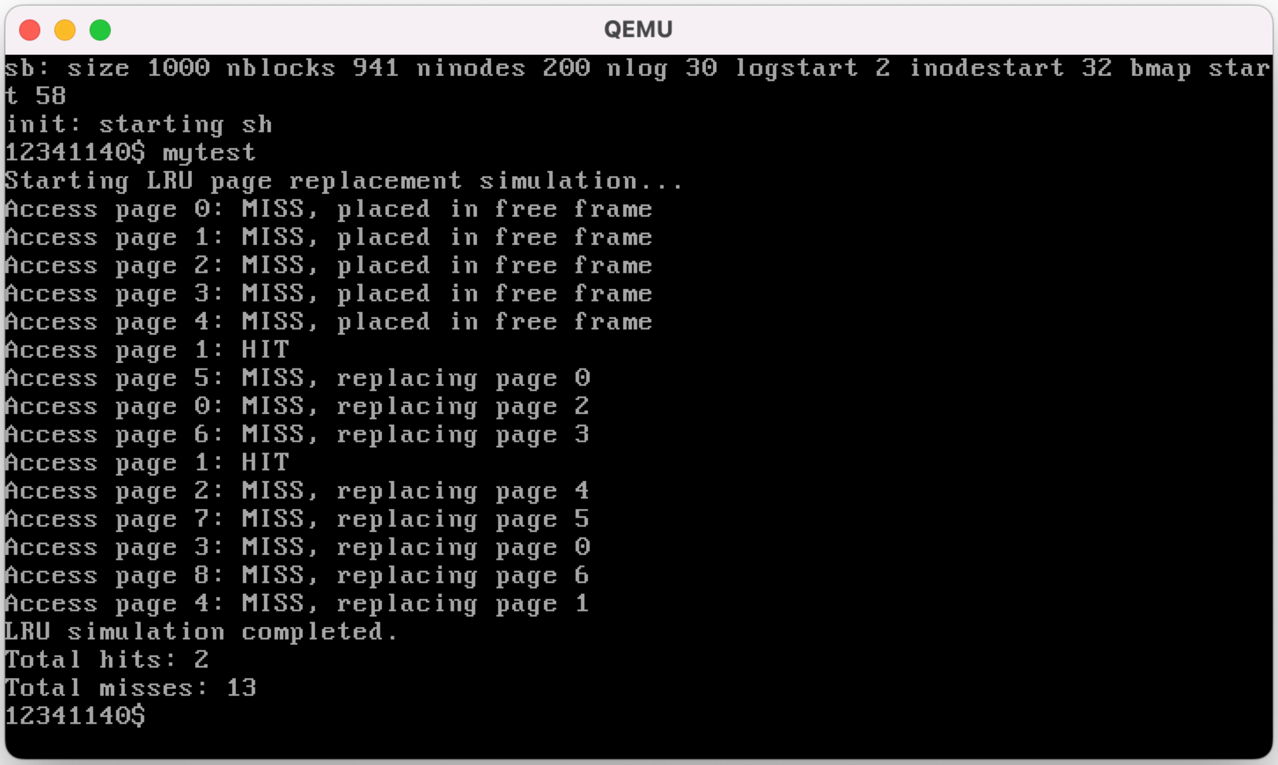
(3) Changed mytest.c to the following:

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

#define MAX_PAGES 5
#define TOTAL_ACCESSES 15

int main(int argc, char *argv[]) {
  int lru[MAX_PAGES];
  int 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 replaced = -1;
    for (j = 0; j < MAX_PAGES; j++) {
      if (lru[j] == -1) {
        lru[j] = page;
        last_used[j] = time;
        replaced = j;
        break;
      }
    }

    if (replaced == -1) {
      int lru_index = 0, min_time = last_used[0];
      for (j = 1; j < MAX_PAGES; j++) {
        if (last_used[j] < min_time) {
          min_time = last_used[j];
          lru_index = j;
        }
      }
      printf(1, "Access page %d: MISS, replacing page %d\n", page, lru[lru_index]);
      lru[lru_index] = page;
      last_used[lru_index] = time;
    } else {
      printf(1, "Access page %d: MISS, placed in free frame\n", page);
    }
  }
  }

  printf(1, "LRU simulation completed.\n");
  printf(1, "Total hits: %d\n", hit);
  printf(1, "Total misses: %d\n", miss);
  exit();
}
```

(4) Inside UPROGS in Makefile , the following was already present:

_mytest



```
QEMU
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
init: starting sh
12341140$ mytest
Starting LRU page replacement simulation...
Access page 0: MISS, placed in free frame
Access page 1: MISS, placed in free frame
Access page 2: MISS, placed in free frame
Access page 3: MISS, placed in free frame
Access page 4: MISS, placed in free frame
Access page 1: HIT
Access page 5: MISS, replacing page 0
Access page 0: MISS, replacing page 2
Access page 6: MISS, replacing page 3
Access page 1: HIT
Access page 2: MISS, replacing page 4
Access page 7: MISS, replacing page 5
Access page 3: MISS, replacing page 0
Access page 8: MISS, replacing page 6
Access page 4: MISS, replacing page 1
LRU simulation completed.
Total hits: 2
Total misses: 13
12341140$
```

Screenshot of the QEMU Terminal

Changes made for xv6-public-FIFO:

(1) Modified proc.h:

```
#define MAX_PAGES 20
```

Also added in struct proc:

```
  int pages[MAX_PAGES];   // FIFO pages
  int page_count;         // number of active frames
```

(2) In vm.c, modified allocuvm():

```
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) {
      kfree(mem);
      deallocuvm(pgdir, newsz, oldsz);
      return 0;
    }

    struct proc *cur = myproc();
    if (cur) {
      if (cur->page_count < MAX_PAGES) {
        cur->pages[cur->page_count++] = (int)a;
        cprintf("Allocated page %d at VA 0x%x\n", cur->page_count, a);
      } else {
        int evict = cur->pages[0];
        // Free victim frame
        char *victim_mem = (char*)P2V(PTE_ADDR(*walkpgdir(pgdir, (void*)evict, 0)));
        kfree(victim_mem);

        // Shift all pages left
        for (int i = 1; i < MAX_PAGES; i++)
          cur->pages[i-1] = cur->pages[i];
        // Put new page at the end
        cur->pages[MAX_PAGES-1] = (int)a;

        cprintf("Evicted page at 0x%x, allocated new page at 0x%x\n", evict, a);
      }
    }
  }
  return newsz;
}
```

(3) Changed mytest.c to the following:

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

#define TOTAL_ACCESSES 15

int main(int argc, char *argv[]) {
  if (argc < 2) {
    printf(1, "Usage: mytest <frames>\n");
    exit();
  }

  int MAX_PAGES = atoi(argv[1]);   // frame size from command line
  int fifo[MAX_PAGES];
  int next_to_replace = 0;
  int i, j, page, hit = 0, miss = 0;

  for (i = 0; i < MAX_PAGES; i++)
    fifo[i] = -1;

  printf(1, "Starting FIFO page replacement simulation (frames=%d)...\n", MAX_PAGES);

  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];
    int found = 0;

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

    if (found) {
      hit++;
      printf(1, "Access page %d: HIT\n", page);
    } else {
      miss++;
      printf(1, "Access page %d: MISS, replacing page %d\n", page, fifo[next_to_replace]);
      fifo[next_to_replace] = page;
      next_to_replace = (next_to_replace + 1) % MAX_PAGES;
    }
  }

  printf(1, "FIFO simulation completed.\n");
  printf(1, "Total hits: %d\n", hit);
  printf(1, "Total misses: %d\n", miss);
  exit();
}
```

(4) Inside UPROGS in Makefile , the following was already present:

```
_mytest
```

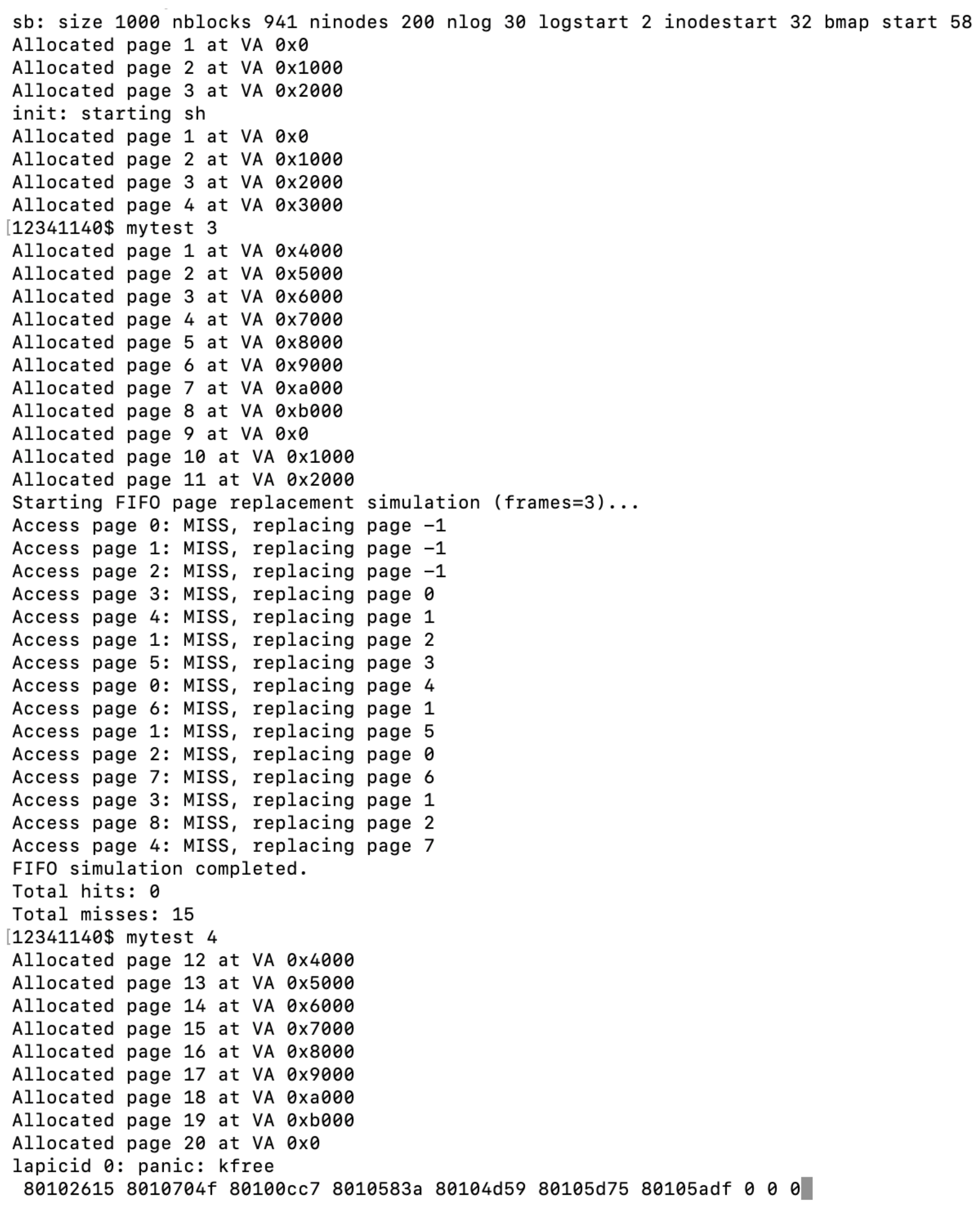
Observation about Belady's Anomaly (FIFO)

Frames = 3 : 0 hits, 15 misses

Frames = 4 : simulation crashed due to kfree panic, results not recorded

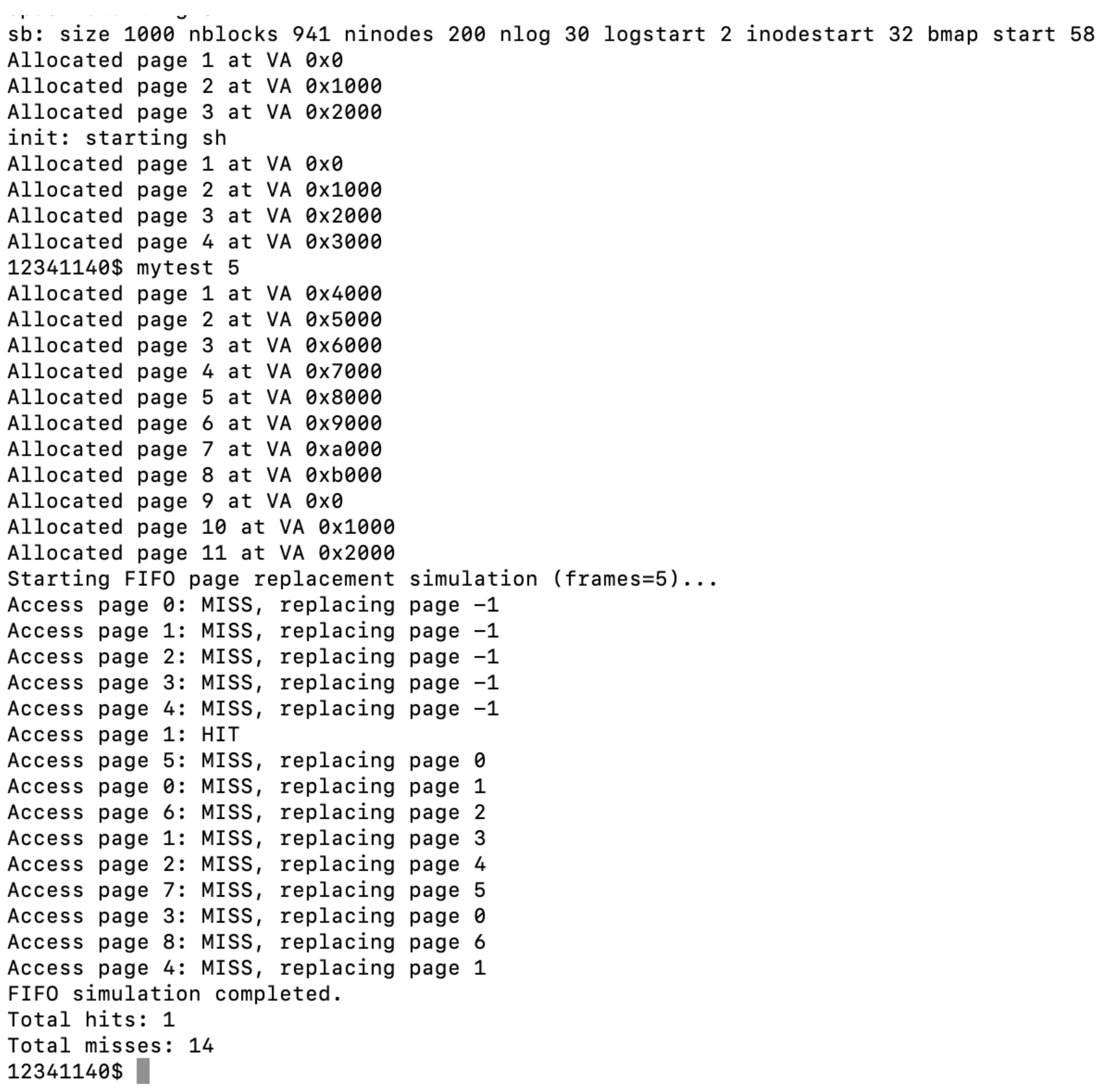
Frames = 5 : 1 hit, 14 misses

From the results, we see that increasing the number of frames from 3 to 5 reduces the number of misses (15 to 14).
This is the expected behavior, as having more frames generally improves performance.
Since the miss count decreased with more frames, Belady's anomaly was not observed in this experiment.
The crash at 4 frames limits full verification, but the available data supports normal FIFO behavior.



```
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
Allocated page 1 at VA 0x0
Allocated page 2 at VA 0x1000
Allocated page 3 at VA 0x2000
init: starting sh
Allocated page 1 at VA 0x0
Allocated page 2 at VA 0x1000
Allocated page 3 at VA 0x2000
Allocated page 4 at VA 0x3000
12341140$ mytest 3
Allocated page 1 at VA 0x4000
Allocated page 2 at VA 0x5000
Allocated page 3 at VA 0x6000
Allocated page 4 at VA 0x7000
Allocated page 5 at VA 0x8000
Allocated page 6 at VA 0x9000
Allocated page 7 at VA 0xa000
Allocated page 8 at VA 0xb000
Allocated page 9 at VA 0x0
Allocated page 10 at VA 0x1000
Allocated page 11 at VA 0x2000
Starting FIFO page replacement simulation (frames=3)...
Access page 0: MISS, replacing page -1
Access page 1: MISS, replacing page -1
Access page 2: MISS, replacing page -1
Access page 3: MISS, replacing page 0
Access page 4: MISS, replacing page 1
Access page 1: MISS, replacing page 2
Access page 5: MISS, replacing page 3
Access page 0: MISS, replacing page 4
Access page 6: MISS, replacing page 1
Access page 1: MISS, replacing page 5
Access page 2: MISS, replacing page 0
Access page 7: MISS, replacing page 6
Access page 3: MISS, replacing page 1
Access page 8: MISS, replacing page 2
Access page 4: MISS, replacing page 7
FIFO simulation completed.
Total hits: 0
Total misses: 15
12341140$ mytest 4
Allocated page 12 at VA 0x4000
Allocated page 13 at VA 0x5000
Allocated page 14 at VA 0x6000
Allocated page 15 at VA 0x7000
Allocated page 16 at VA 0x8000
Allocated page 17 at VA 0x9000
Allocated page 18 at VA 0xa000
Allocated page 19 at VA 0xb000
Allocated page 20 at VA 0x0
lapicid 0: panic: kfree
 80102615 8010704f 80100cc7 8010583a 80104d59 80105d75 80105adf 0 0 0
```

Screenshot of the Terminal for frame sizes 3 and 4



```
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
Allocated page 1 at VA 0x0
Allocated page 2 at VA 0x1000
Allocated page 3 at VA 0x2000
init: starting sh
Allocated page 1 at VA 0x0
Allocated page 2 at VA 0x1000
Allocated page 3 at VA 0x2000
Allocated page 4 at VA 0x3000
12341140$ mytest 5
Allocated page 1 at VA 0x4000
Allocated page 2 at VA 0x5000
Allocated page 3 at VA 0x6000
Allocated page 4 at VA 0x7000
Allocated page 5 at VA 0x8000
Allocated page 6 at VA 0x9000
Allocated page 7 at VA 0xa000
Allocated page 8 at VA 0xb000
Allocated page 9 at VA 0x0
Allocated page 10 at VA 0x1000
Allocated page 11 at VA 0x2000
Starting FIFO page replacement simulation (frames=5)...
Access page 0: MISS, replacing page -1
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 5: MISS, replacing page 0
Access page 0: MISS, replacing page 1
Access page 6: MISS, replacing page 2
Access page 1: MISS, replacing page 3
Access page 2: MISS, replacing page 4
Access page 7: MISS, replacing page 5
Access page 3: MISS, replacing page 0
Access page 8: MISS, replacing page 6
Access page 4: MISS, replacing page 1
FIFO simulation completed.
Total hits: 1
Total misses: 14
12341140$ 
```

Screenshot of the Terminal for frame size 5