Assignment 3

What you have to do

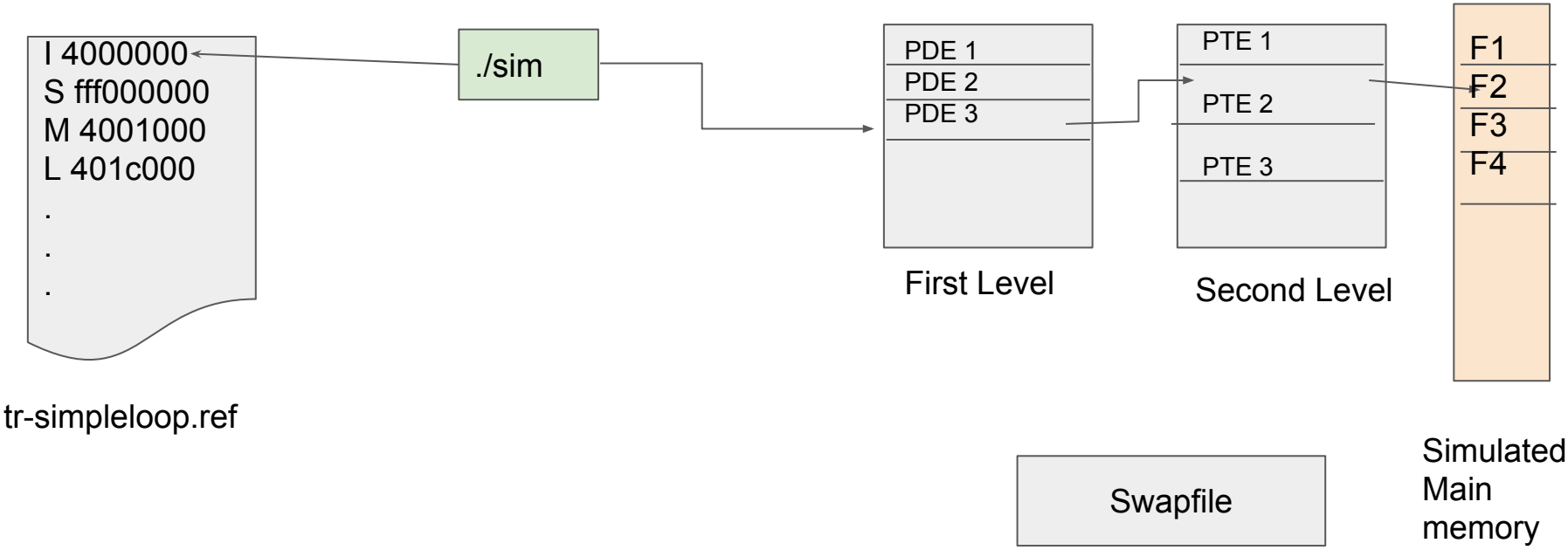
1. **Implement address translation** for a virtual address, using two level page table
2. Implement **page replacement** algorithms
3. Prepare a **report** based on experimentation

Code workflow

```
./sim -f ./traceprogs/tr-simpleloop.ref -m 50 -s 3000 -a rand
```

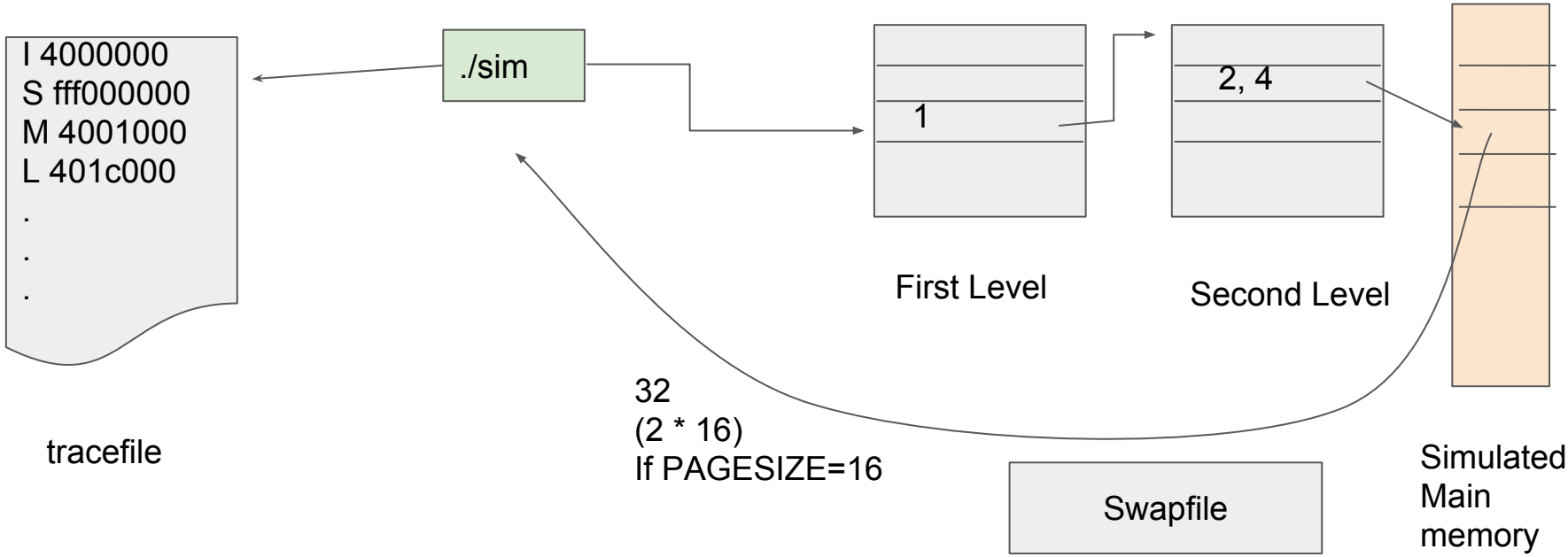
Code workflow

```
sim -f ./traceprogs/tr-simpleloop.ref -m 50 -s 3000 -a rand
```



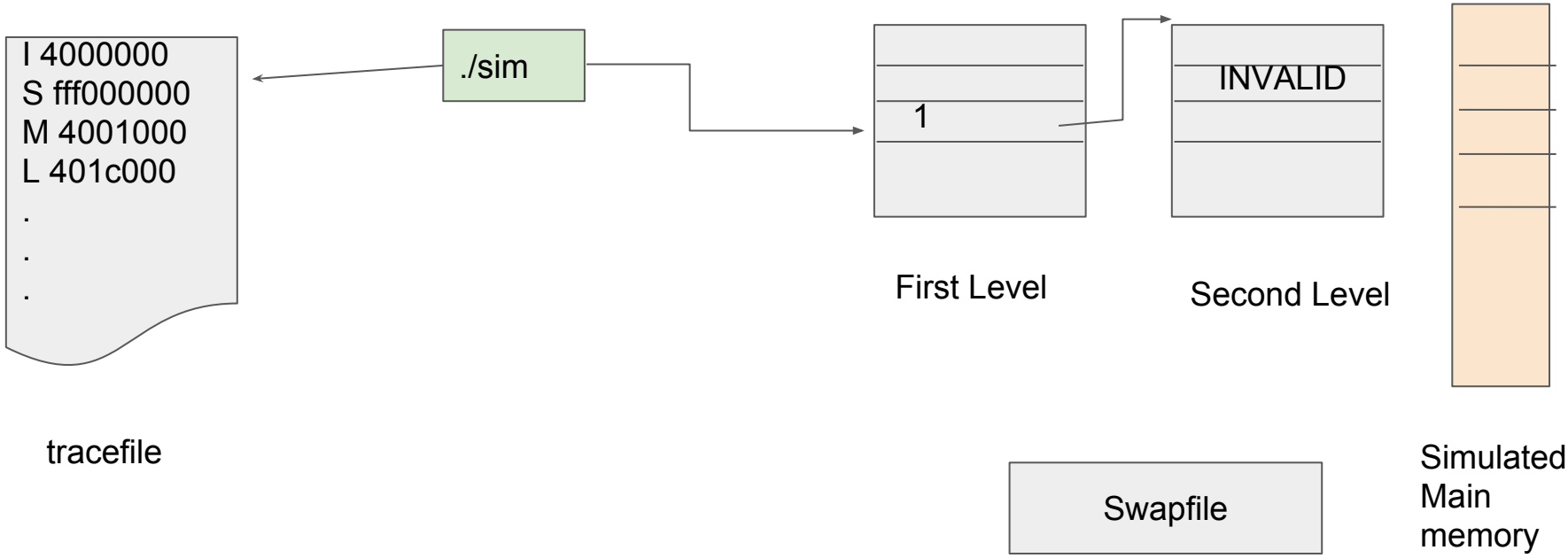
Code workflow: Hit

```
sim -f ./traceprogs/tr-simpleloop.ref -m 50 -s 3000 -a rand
```



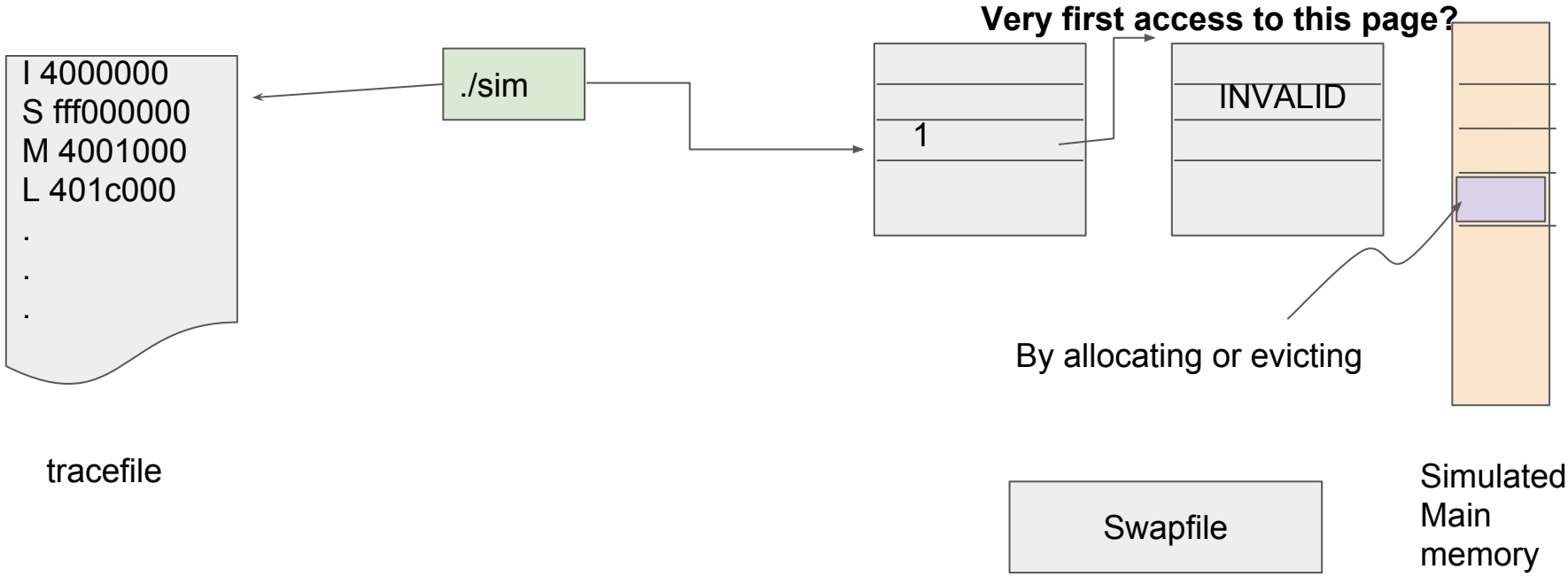
Code workflow: Miss

```
sim -f ./traceprogs/tr-simpleloop.ref -m 50 -s 3000 -a rand
```



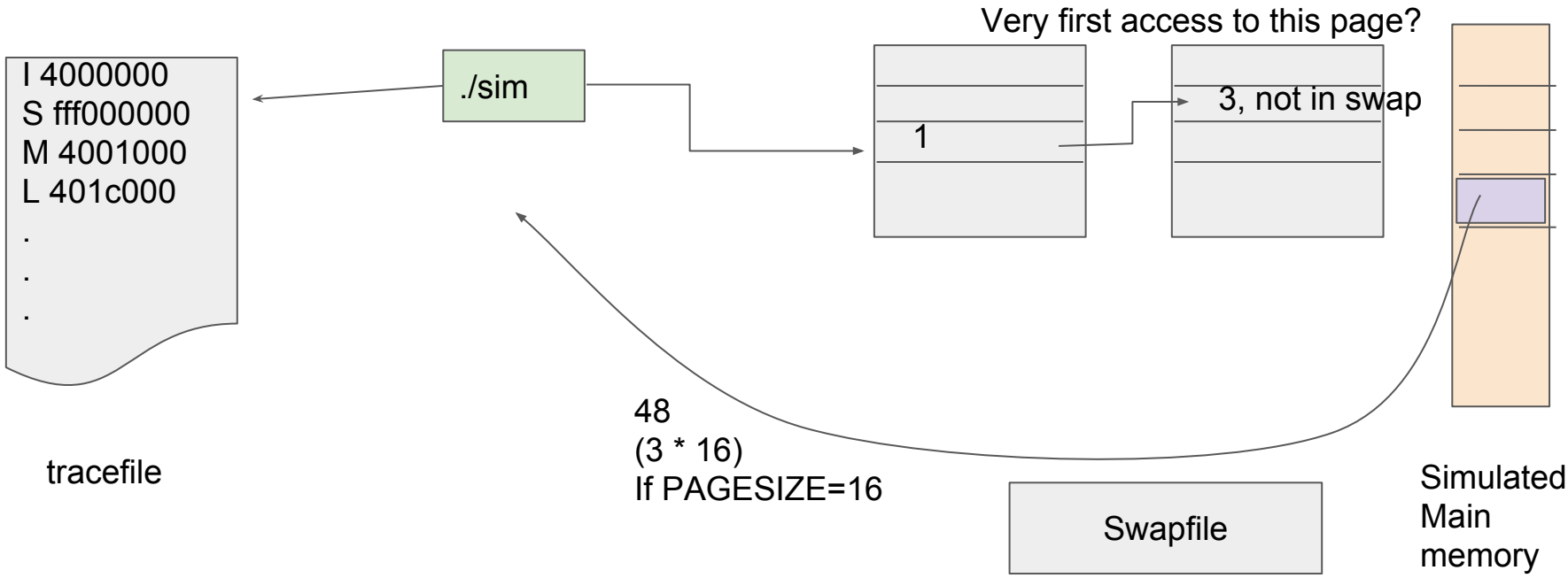
Code workflow: Miss

```
sim -f ./traceprogs/tr-simpleloop.ref -m 50 -s 3000 -a rand
```



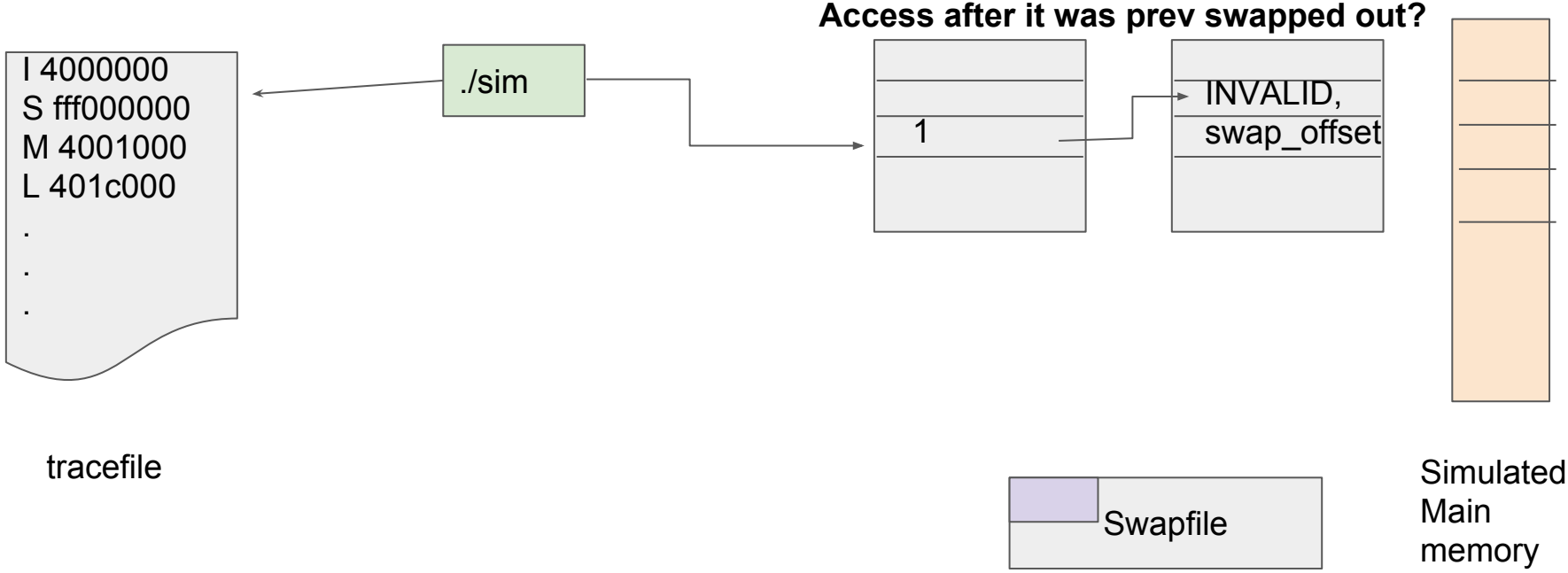
Code workflow: Miss

```
sim -f ./traceprogs/tr-simpleloop.ref -m 50 -s 3000 -a rand
```



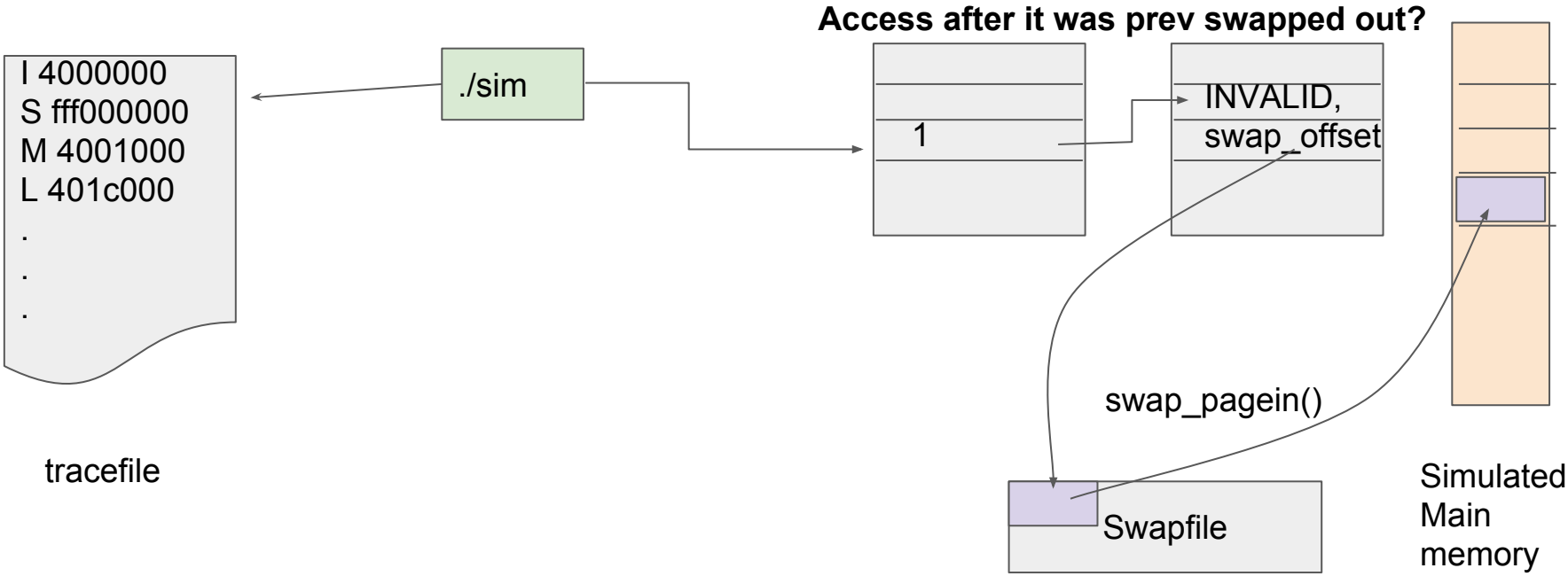
Code workflow: Miss

```
sim -f ./traceprogs/tr-simpleloop.ref -m 50 -s 3000 -a rand
```



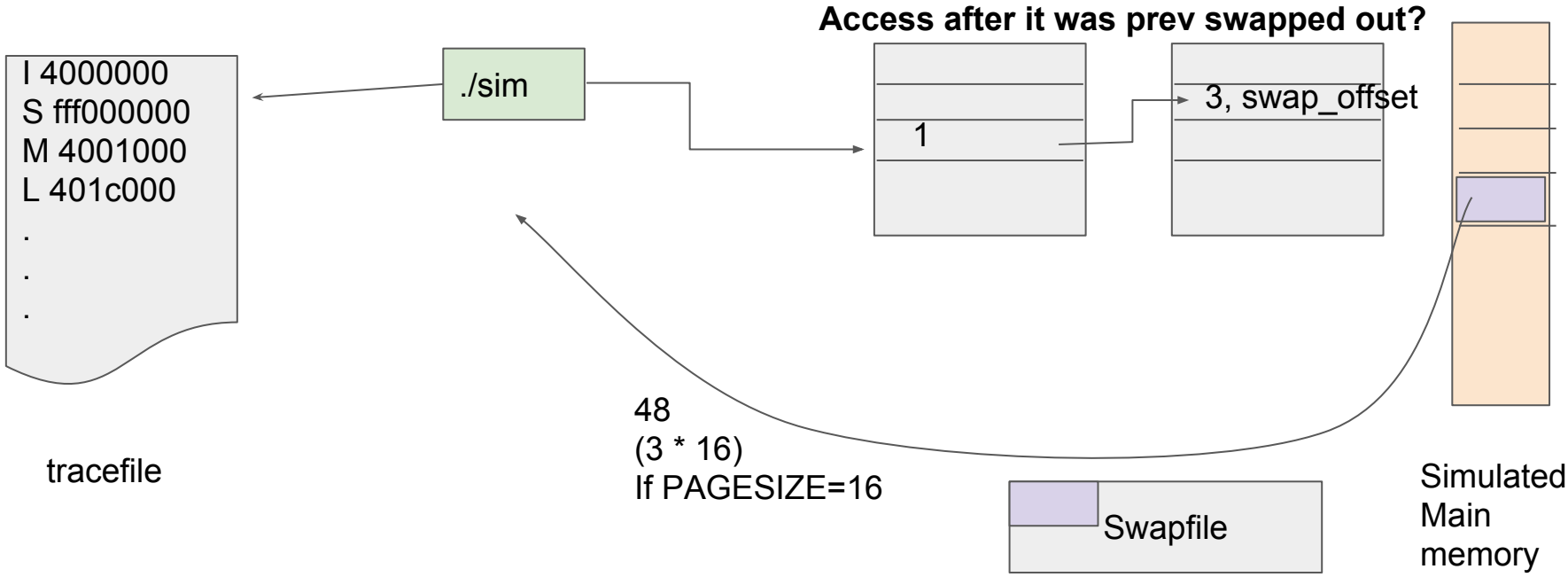
Code workflow: Miss

```
sim -f ./traceprogs/tr-simpleloop.ref -m 50 -s 3000 -a rand
```



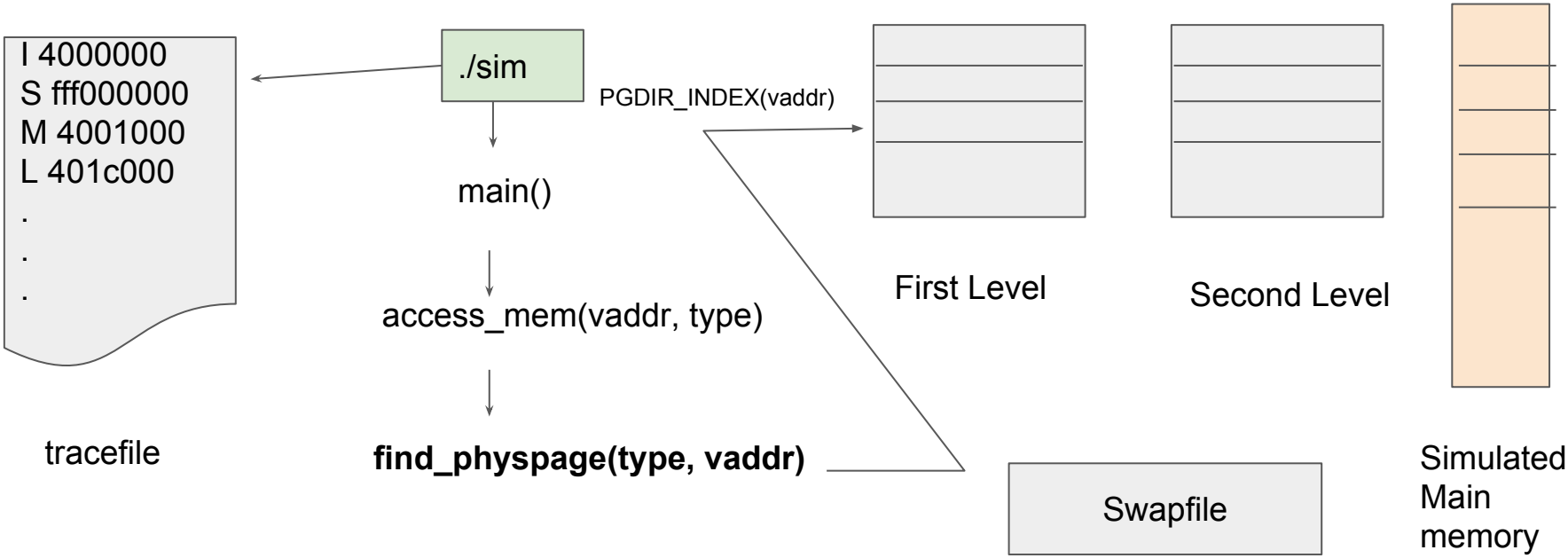
Code workflow: Miss

```
sim -f ./traceprogs/tr-simpleloop.ref -m 50 -s 3000 -a rand
```



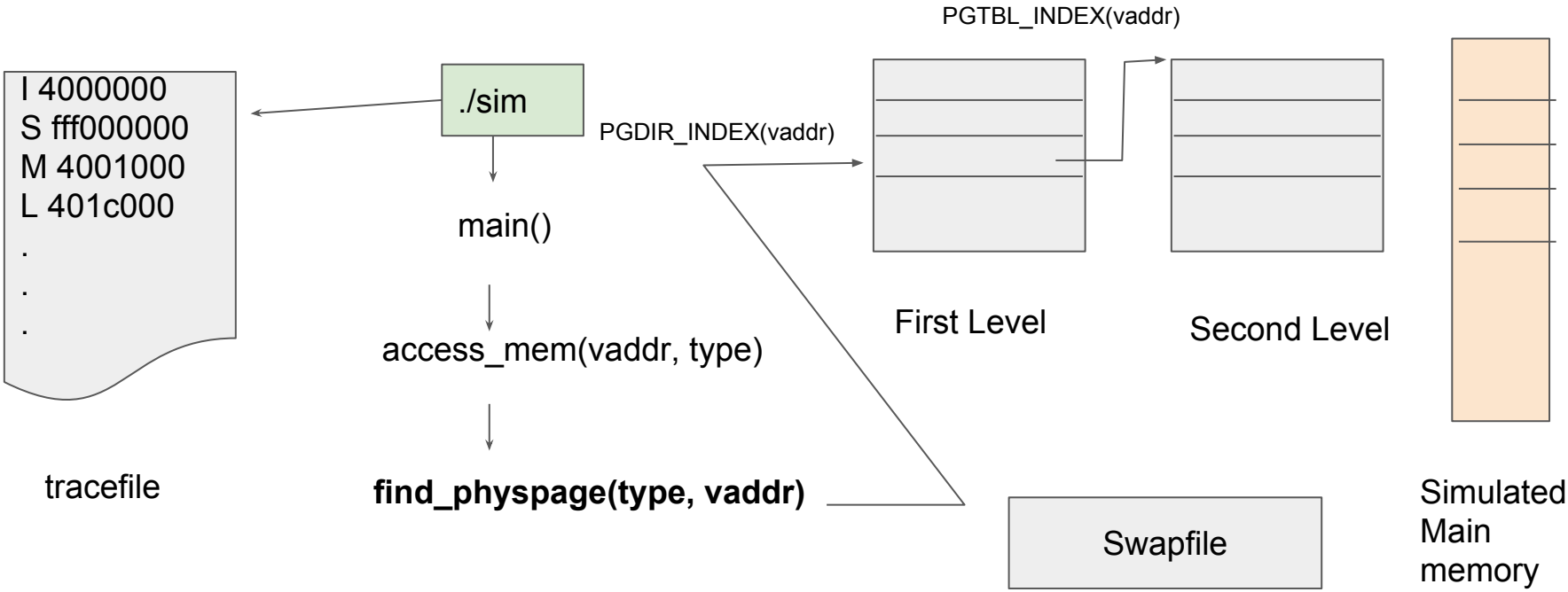
Some functions and macros

```
sim -f ./traceprogs/tr-simpleloop.ref -m 50 -s 3000 -a rand
```



Some functions and macros

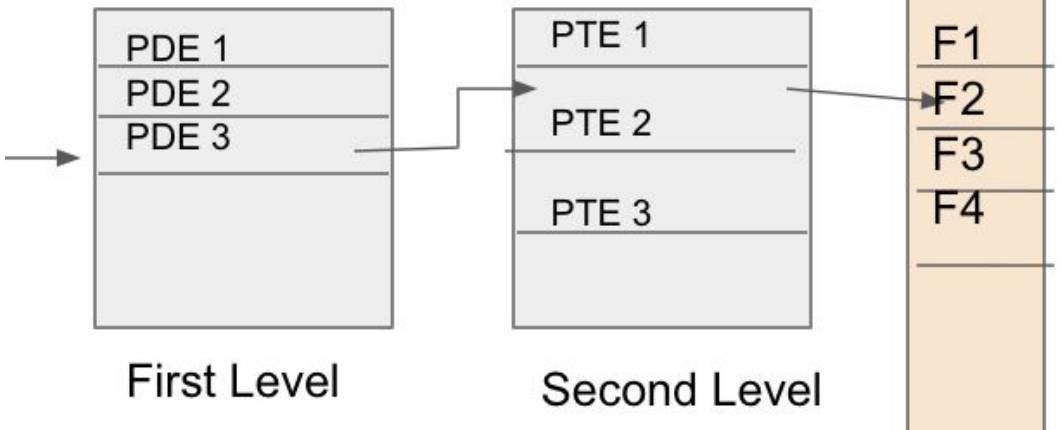
```
sim -f ./traceprogs/tr-simpleloop.ref -m 50 -s 3000 -a rand
```



Data Structures

char ***physmem** = malloc(memsize * SIMPAGESIZE)

pgdir_entry_t **pgdir**[PTRS_PER_PGDIR];

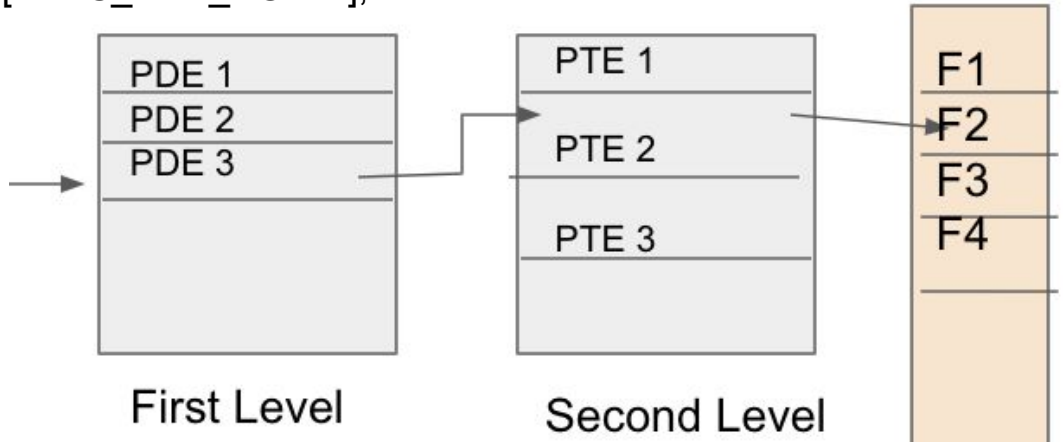


pgtbl_entry_t ***pgtbl**;
//array of pgtble_entry_t

Data Structures

char ***physmem** = malloc(memsize * SIMPAGESIZE)

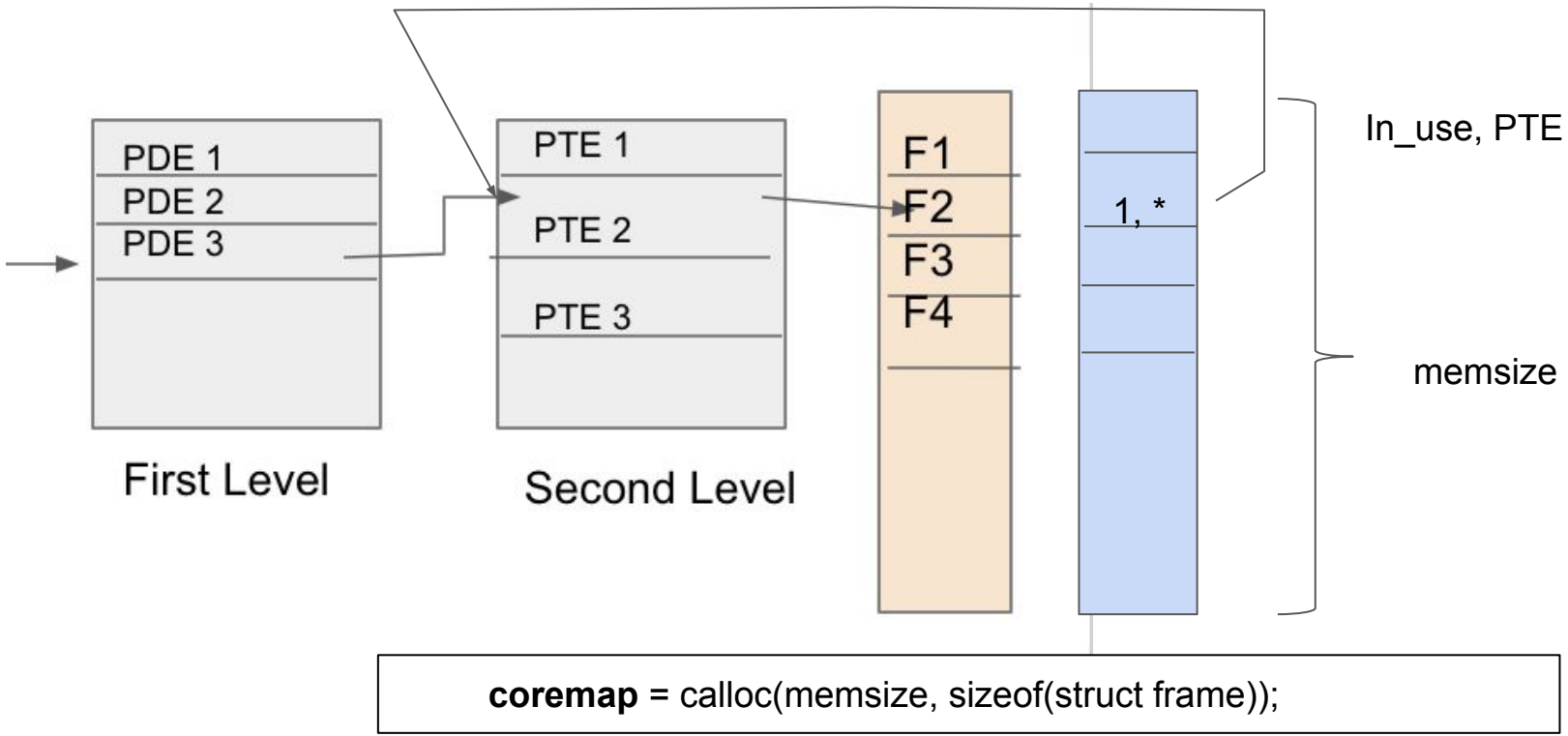
pgdir_entry_t **pgdir**[PTRS_PER_PGDIR];



pgtbl_entry_t ***pgtbl**;
//array of pgtble_entry_t

```
unsigned idx = PGDIR_INDEX(vaddr);
pgtbl_entry_t *pgtbl = (pgtbl_entry_t *)(pgdir[idx].pde & ~PG_VALID);
```

Other data structures



Other functions

int allocate_frame(pgtbl_entry_t *p)

- Finds a physical frame number for the page p
- evict if it has to

swap_pagein()

swap_pageout()

Bring page in and out of swapfile.

Page Replacement Algorithms

```
extern void rand_init();
extern void lru_init();
extern void clock_init();
extern void fifo_init();
extern void opt_init();

// These may not need to do anything for some algorithms
extern void rand_ref(pgtbl_entry_t *);
extern void lru_ref(pgtbl_entry_t *);
extern void clock_ref(pgtbl_entry_t *);
extern void fifo_ref(pgtbl_entry_t *);
extern void opt_ref(pgtbl_entry_t *);

extern int rand_evict();
extern int lru_evict();
extern int clock_evict();
extern int fifo_evict();
extern int opt_evict();
```

Initialization functions: maybe set clock hand to its initial position, maybe do nothing at all, or maybe do some magic for opt

As we saw: called on every reference to a page that is **in the core map**, i.e., in physical memory.
For instance: LRU might want to update “the time” of the last access to the page being referenced.

If the core map is full, these will make space for a new page: returns the index of the frame to be evicted.
For instance: OPT will look into the future and see which of the pages in the coremap will be referenced latest.

Good luck!