

## Simplethreads API

```
void pthread_init()
    * Initialize the whole system
pthread_t pthread_create(func start_func,
    void *arg)
    * Create a new thread and make it runnable
void pthread_yield()
    * Give up the CPU
void pthread_exit(void *ret)
    * Exit current thread
void* pthread_join(pthread_t t)
    * Wait for specified thread to exit
```

10/21/10

1

## Simplethreads internals

### \* Structure of the TCB:

```
struct _pthread {
    pthread_ctx_t *saved_ctx;
    /**
     * Add your fields to the thread
     * data structure here.
     */
};
```

10/21/10

2

## Sample multithreaded program

```
int main(int argc, char **argv) {
    pthread_init();
    for(i = 0; i < 3; i++) {
        if (pthread_create(thread_start,
            (void *)i) == NULL) {
            printf("pthread_create failed\n");
            exit(1);
        }
    }
    pthread_yield();
    printf("back in main\n");
    return 0;
}
```

10/21/10

4

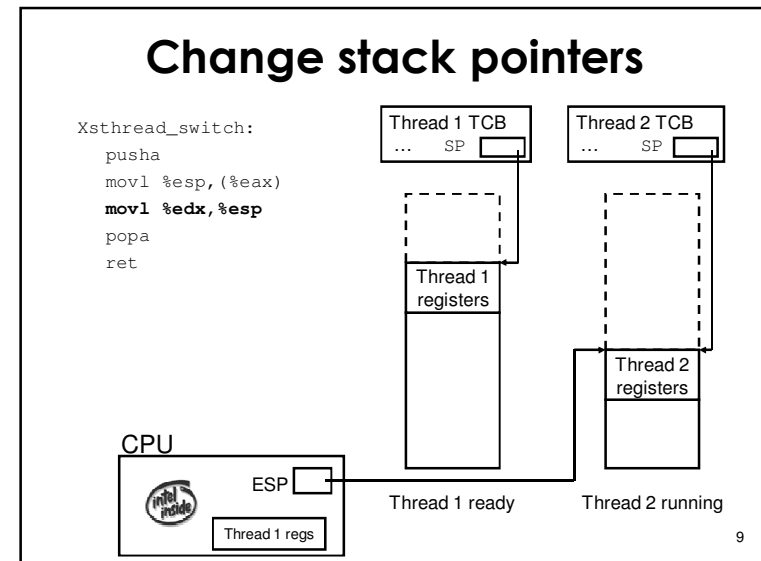
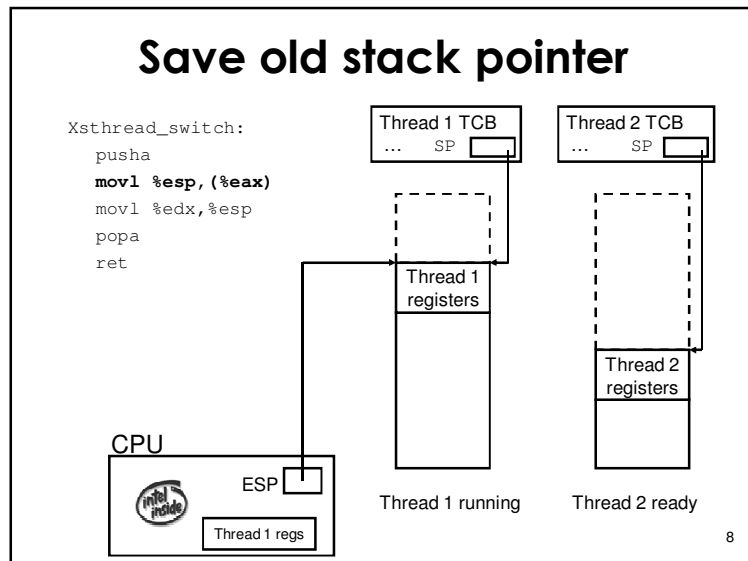
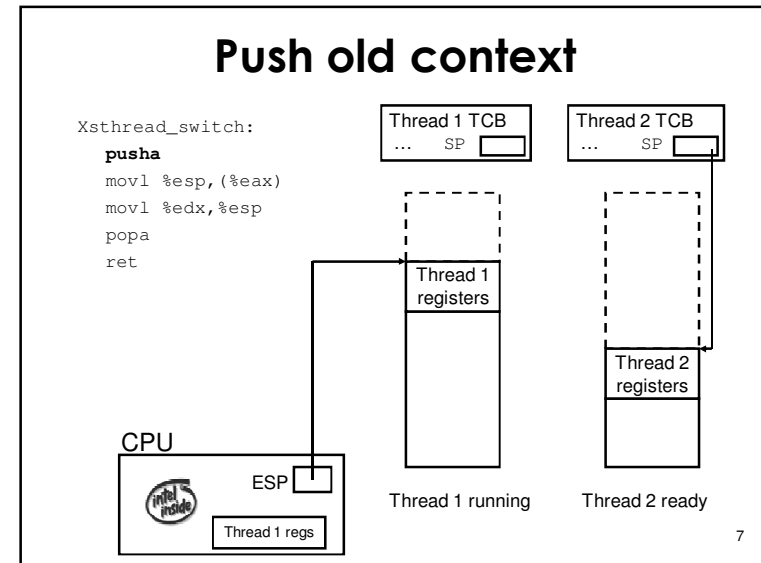
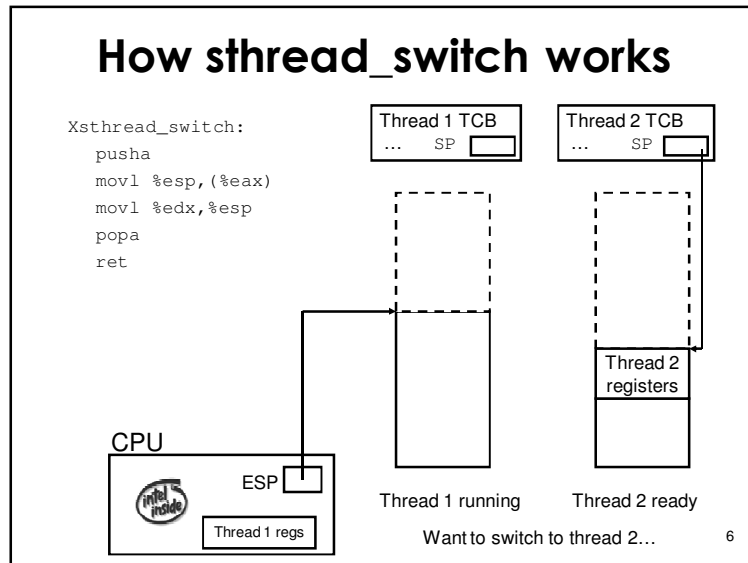
## Managing contexts

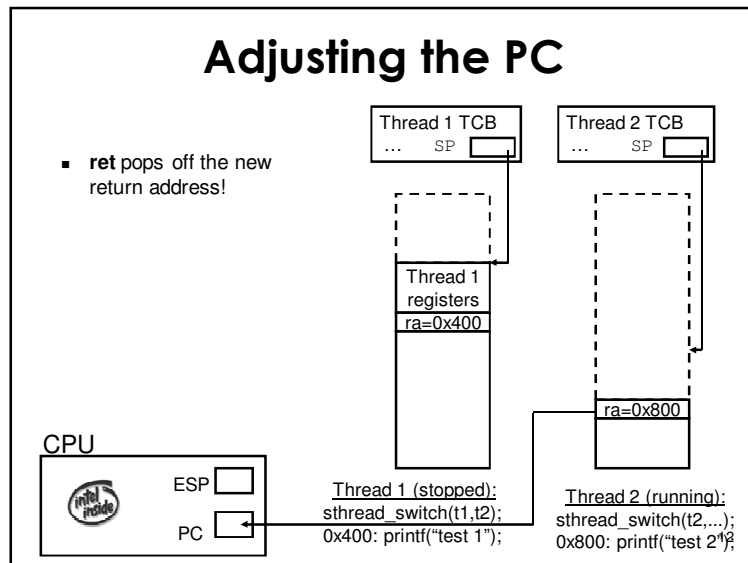
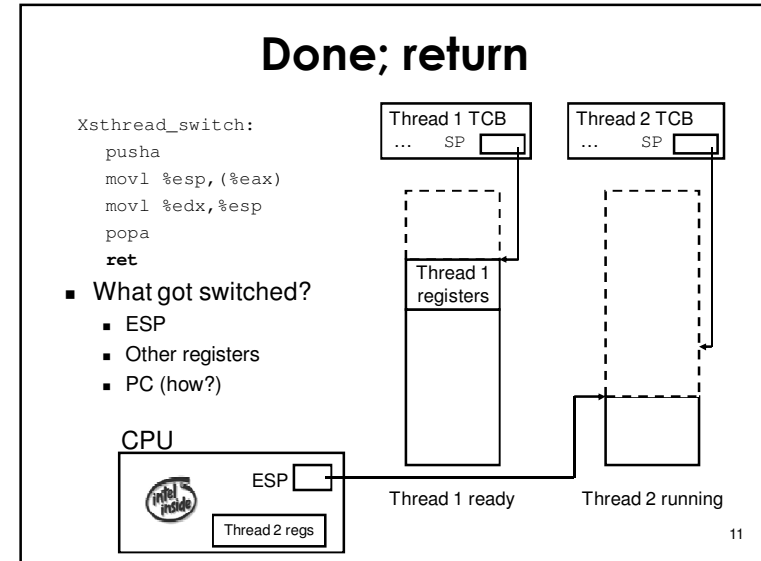
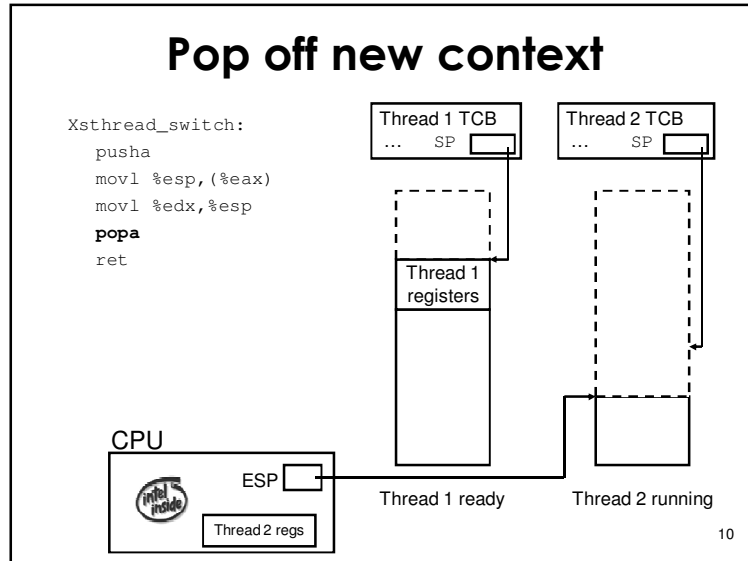
### \* Thread *context* = thread stack + stack pointer

```
pthread_new_ctx(func_to_run)
    * creates a new thread context that can be switched to
pthread_free_ctx(some_old_ctx)
    * Deletes the supplied context
pthread_switch(oldctx, newctx)
    * Puts current context into oldctx
    * Takes newctx and makes it current
```

10/21/10

5





### Trabalho para casa

1. Descobrir como acontece o despacho periódico nas sthreads
2. Perceber onde é feito o escalonamento round-robin das sthreads

(Discutiremos vossas conclusões na próxima aula teórica.)

2007 Sistemas Operativos