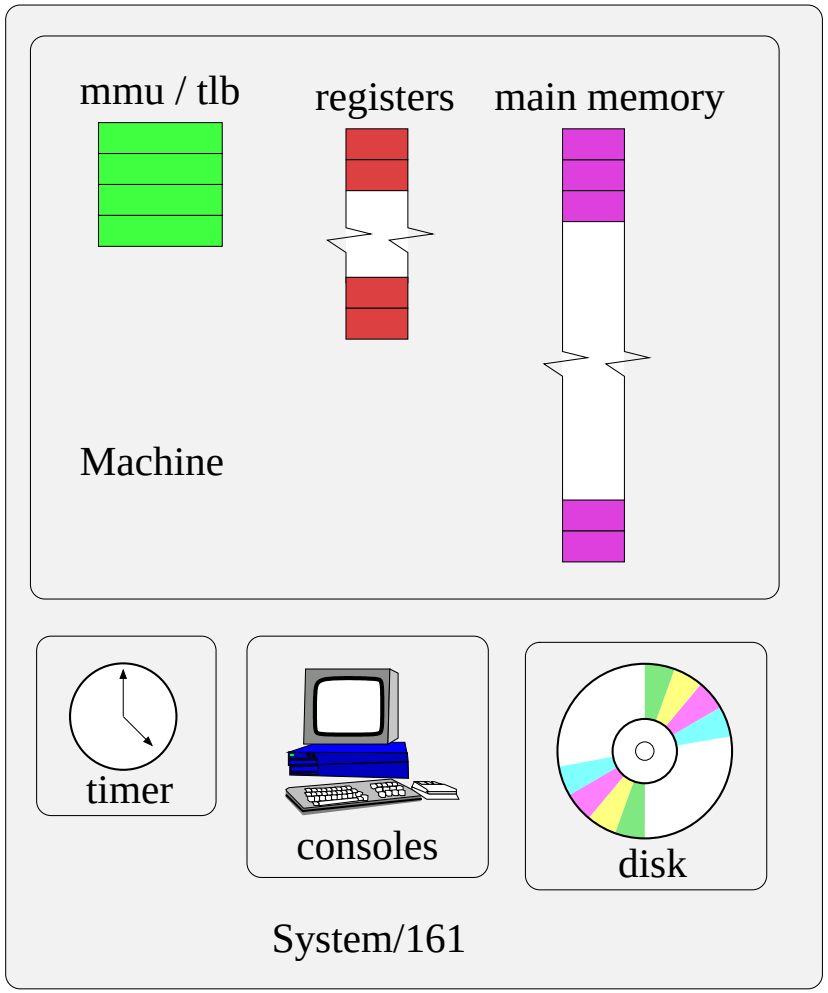
What is OS/161?

workstation simulator: the simulated workstation is implemented in System/161 and includes a MIPS processor (R3000 in big-endian mode), main memory, and a collection of devices including a timer, disk(s), a network interface, and input and output consoles.

operating system: the OS/161 operating system runs on System/161 and manages the simulated workstation, and implements a set of system calls for user programs

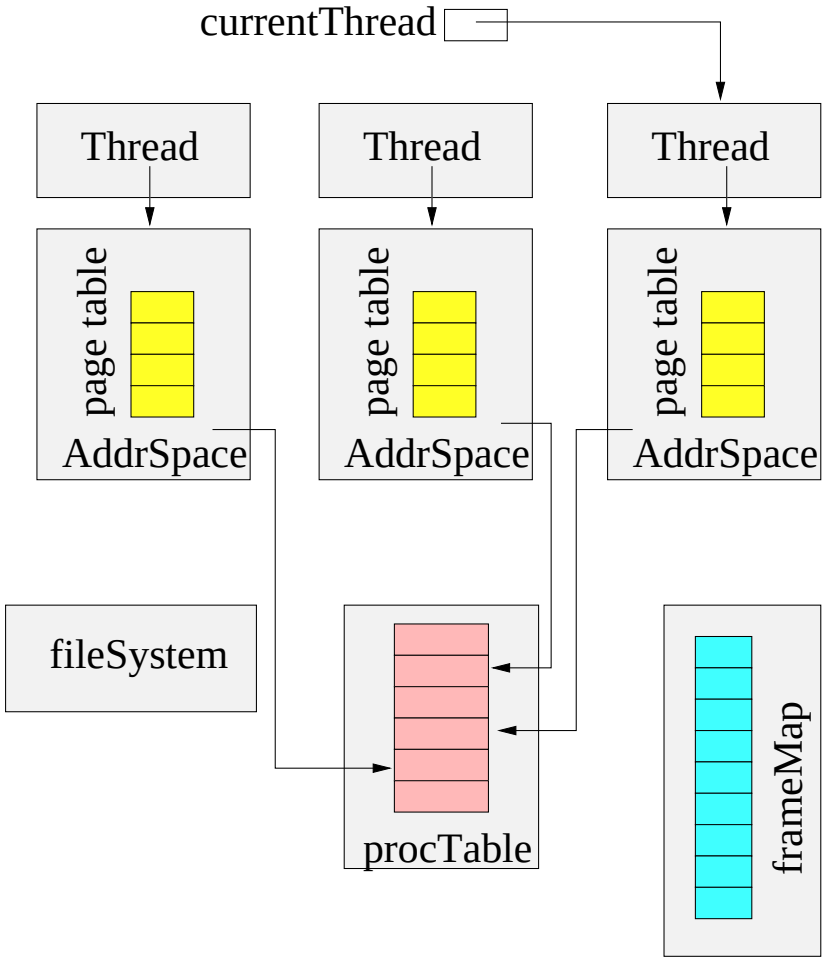
user programs: OS/161 user programs run on the simulated machine, and use services provided by the OS/161 operating system

The Systems/161 Machine Simulator



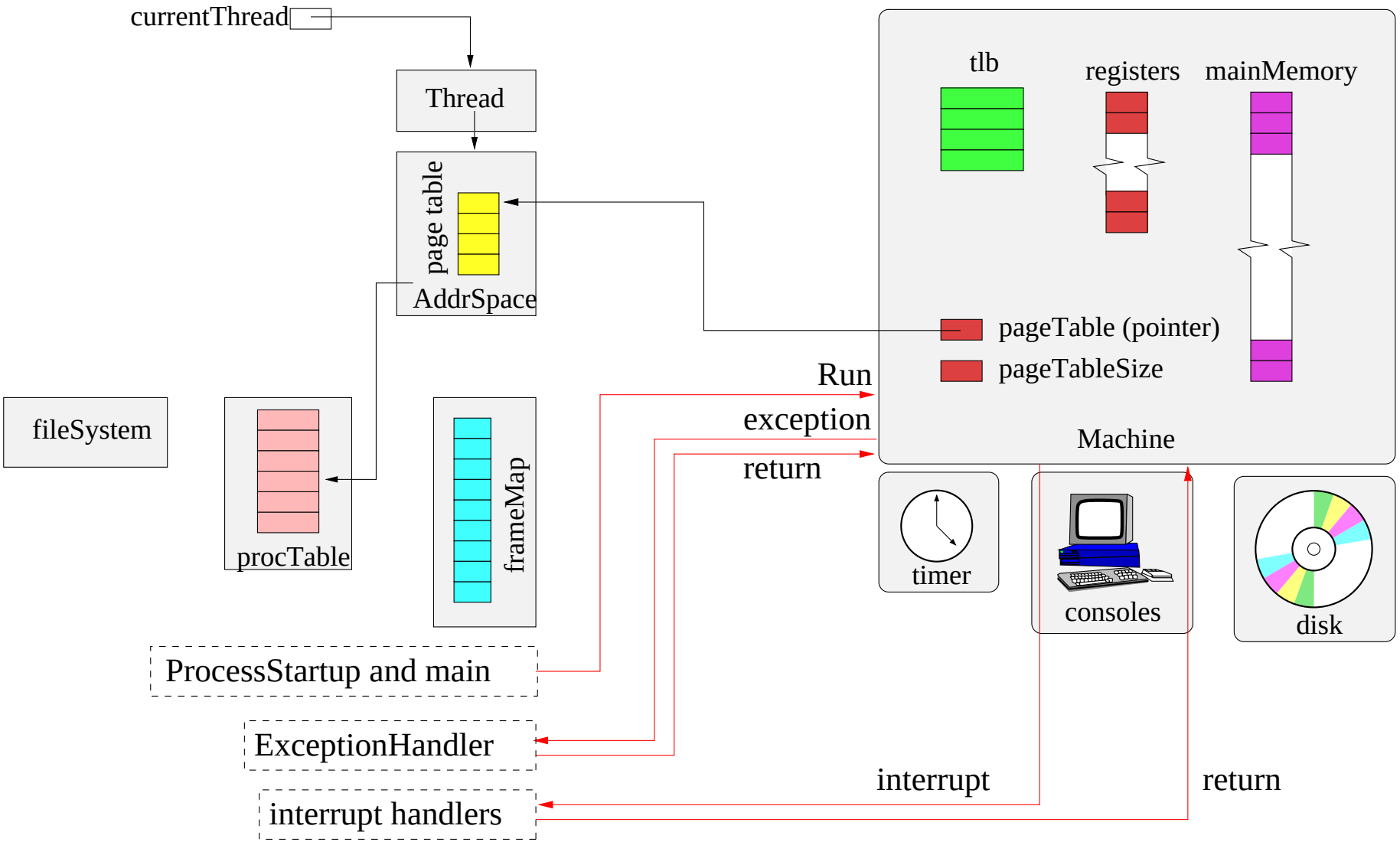
- registers include the program counter and stack pointer
- main memory consists of `ramsize=524288` bytes (from sys161.conf), the frame size is
  `#define PAGE_SIZE 4096`
- some devices (e.g., network, second disk) are not shown
- simulator include the MMU (coprocessor 0) which implements a TLB.

Some Components of The NachOS Kernel



- not all OS components are shown
- each NachOS process has an entry in `ProcTable`, a thread, and an address space
- address spaces are implemented by `AddrSpace` objects, which include a page table
- `frameMap` tracks which main memory frames are in use
- NachOS has two file system implementations

Summary of Machine/Kernel Interactions in NachOS



NachOS Thread Facilities

Threads: new threads can be created, and threads can be destroyed. Each new thread executes a kernel procedure that is specified when the thread is created. (`threads/thread.*`)

Scheduling: a round-robin ready queue for threads (`threads/scheduler.*`)

Synchronization: semaphores, locks, and condition variables. These are integrated with the scheduler: blocked threads are kept off of the ready queue, unblocked threads are placed back onto the ready queue. (`threads/synch.*`)

NachOS Workstation Devices

- like many real devices, the NachOS workstation's simulated devices are *asynchronous*, which means that they use interrupts to notify the kernel that a requested operation has been completed, or that a new operation is possible. For example:
  - the input console (keyboard) generates an interrupt each time a new input character is available
  - the output console (display) can only output one character at a time. It generates an interrupt when it is ready to accept another character for output.
  - the disk accepts one read/write request at a time. It generates an interrupt when the request has been completed.
- the kernel implements *synchronous* interfaces to each of these devices
  - implemented using the synchronization primitives
  - synchronous interfaces are much easier for the rest of the kernel to use than the asynchronous interfaces. Use them!

Example: Synchronous Input Console

- `SynchConsoleInput::GetChar()` returns one character from the console, and causes the calling thread to *block* (until a character is available) if there are no available input characters.
- Implementation uses a single semaphore:
  - `SynchConsoleInput::GetChar()` does a `P()` before attempting to read a character from the input console.
  - Input console interrupt handler does a `V()`

The NachOS Stub File System

- NachOS has two file system implementations.
  - The real file system has very limited functionality. Files are stored on the workstation's simulated disk.
  - The "stub" file system stores files outside of the simulated machine, in the file system of the machine on which NachOS is running. Magic!
- Until Asst 3, the "stub" file system is used. This is why a file that is created by a NachOS user program appears on the machine on which NachOS is running. This is also why NachOS user programs can be stored in files on host machine, and not on the simulated workstation.
- The "stub" file system may seem unrealistic, however, a diskless workstation with network boot uses a similar mechanism.

The NachOS File System: disk.h

```
#define SectorSize      128 // bytes
#define SectorsPerTrack 32
#define NumTracks       64  // per disk
#define NumSectors (SectorsPerTrack * NumTracks)
                   // 32 * 64 = 2048
```

Disk Size = 2048 sectors * 128 bytes = 256 KB

The NachOS File System: filesys.h

```
#define FreeMapSector     0
#define DirectorySector   1

#define FreeMapFileSize (NumSectors / BitsInByte)
                        // 2048 / 8 = 256

#define NumDirEntries     10
#define DirectoryFileSize
   (sizeof(DirectoryEntry) * NumDirEntries)
   // 20 * 10 = 200 (1.5625 sectors)
```

The NachOS File System: filesys.h (cont'd)

```
class FileSystem {
        ...
private:
  // Bit map of free disk blocks,
  // represented as a file
  OpenFile* freeMapFile;

  // "Root" directory -- list of
  // file names, represented as a file
  OpenFile* directoryFile;
};
```

FreeMap file has 2048 entries and occupies 2 sectors (256 bytes), plus one sector for its header. Directory file has 10 entries, which requires 200 bytes (2 sectors), plus one sector for its header.

The NachOS File System: filehdr.h

```
#define NumDirect
  ((SectorSize - 2 * sizeof(int))
   / sizeof(int))
#define MaxFileSize
  (NumDirect * SectorSize)

class FileHeader {
  ...
private:
  int numBytes;
  int numSectors;                 // data sectors
  int dataSectors[NumDirect];  // sector numbers
...
```

The NachOS File System: filehdr.h (cont'd)

- `FileHeader` fits in one sector = 128 bytes
- first two fields (`numBytes` and `numSectors`) use 8 bytes
- 120 bytes are left for block pointers
- each block pointer requires 4 bytes, so

$$\texttt{NumDirect} = \frac{128 - 2 * 4}{4} = 30$$

- maximum file size is:

$$MaxFileSize = \texttt{NumDirect} * \texttt{SectorSize} = 30 * 128 = 3840$$

The NachOS File System: directory.h

```
#define FileNameMaxLen 9
// for simplicity, we assume
// file names are <= 9 characters long

class DirectoryEntry {
public:
  bool inUse;
  int sector;
  char name[FileNameMaxLen + 1];
  // Text name for file, with +1 for
  // the trailing '\0'
};
```

4 bytes for `inUse`, 4 bytes for `sector`, 10 bytes for `name`.

File System Command Line Example

```
mobey 1% nachos -f
Machine halting!

Ticks: total 14010, idle 14000, system 10, user 0
Disk I/O: reads 3, writes 6
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
```

File System Command Line Example (part 2)

```
mobey 2% nachos -D
Bit map file header:
FileHeader contents.  File size: 256.  File blocks:
2 3
File contents:
\0\0\0\?\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0
```

File System Command Line Example (part 3)

```
Directory file header:
FileHeader contents.  File size: 200.  File blocks:
4 5
File contents:
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
Bitmap set:
0, 1, 2, 3, 4, 5,
Directory contents:

Machine halting!
```

File System Command Line Example (part 4)

```
mobey 3% cat > File1
Hello
mobey 4% cat > File2
World
mobey 5%  nachos -cp File1 File1
Machine halting!

mobey 6% nachos -cp File2 File2
Machine halting!
```

File System Command Line Example (part 5)

```
mobey 7% nachos -D
Bit map file header:
FileHeader contents.  File size: 256.  File blocks:
2 3
File contents:
\0\0\3\ff\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
```

File System Command Line Example (part 6)

```
Directory file header:
FileHeader contents.  File size: 200.  File blocks:
4 5
File contents:
\1\0\0\0\0\0\0\6File1\0\0\0\0\0\0\0\1\0\0\0\0\0\0\8
File2\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
Bitmap set:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
```

File System Command Line Example (part 7)

```
Directory contents:
Name: File1, Sector: 6
FileHeader contents.  File size: 6.  File blocks:
7
File contents:
Hello\a
Name: File2, Sector: 8
FileHeader contents.  File size: 6.  File blocks:
9
File contents:
World\a

Machine halting!
```