4. UNIX pipes represent all message-passing information as files. This is in no way necessary, but offers advantages over other methods.

Firstly, having this standardised message-passing system encourages compositionality. It becomes easier for a program to output things that are both human-readable (files stored in the file system permanently) and machine-readable (files used temporarily as part of a piping operation). Combined with this, text files are encouraged above other file types, so it is easier to create standard ways for programs to deal with other programs' output (parser generators, for instance).

Also, passing messages as files has advantages in debugging. It is trivial to represent a message as a permanent file, then inspect it using standard tools.

However, pipes can bring overhead costs. A naive implementation of temporary files would involve writing and reading of the hard disk, which is a very slow operation. Instead, temporary files are stored in a cache, which must be made to act as if it were part of the hard disk.

Also, the preference for human-readable text files often makes parsing output unnecessarily difficult. It is also prone to encoding problems when non-ASCII text

is used.

5 a A typical GUI will provide at least a way of launching applications whilst another application is running. Applications should not be able to arbitrarily modify the desktop environment, so should be kept in a separate process. For new applications to be launched whilst another is running, the desktop environment and the applications must be running (from the user's perspective) simultaneously.

b. For scheduling, processes must be held in some kind of queue. For all but the simplest scheduling algorithms, this will be a priority queue. Usually, a priority queue will be implemented as a heap, where the top element represents the process to be run next. The heap should support fast key updates, since dynamic priorities will change frequently. Static priorities usually don't work well, since some jobs never run.

Alternatively, the simpler scheme of round-robin scheduling could be used. Since this schedules processes in a very predictable way, a cyclic queue can be used. This limits the number of processes allowed, but has better performance and less overhead than a heap.

c. Shortest job first scheduling can be implemented using a minheap in which the keys are the estimated burst lengths. In order to get the

information required to estimate burst lengths, the burst length of the current process has to be measured. When it is finished, the updated estimate has to be propagated through the heap to all similar idle processes. Once the heap is made valid again, the process at the top is picked to be run.

d. When being input or output, continuous media require sufficient CPU time throughout to avoid breakages. They also tend to cause very long burst times, so scheduling algorithms that choose processes with short burst times will select against them.

6 a. To avoid interference between processes, the OS needs to keep the memory of each process separate. To work efficiently, this requires hardware support either for mapping logical addresses to physical addresses (in non-paging systems) or maintaining a TLB (in paging systems). A TLB allows memory addressing to avoid page tables, instead using a quicker lookup table containing just the most recently used addresses.

To avoid locking up the system, the scheduler should be preempive. This will allow it to pause long-running jobs and give the user control over whether they should continue. The scheduler will require a way to time CPU bursts in order to recognise when a process has

been running continuously for too long.

b. The OS will provide a set of control codes, each providing some interaction with the OS. These will often be about the file system, and cause the calling process to become idle until the action has been completed.

c. The hardware optimisations for address translation are not necessary, and can be replicated in software. Without a timer, a simple scheduler, such as round-robin, should be used, since it does not require timing information.

7. a i Byte 70 is in the immediate data, so no further disk blocks need to be read.

ii The direct block stores 2^8 pointers, and each block contains 2^{12} bytes. Hence, the immediate data and direct blocks cover $2^{20} + 2040$ bytes, and byte $2^{20} + 2044$ is in the indirect block.

The indirect block pointer is dereferenced to lead to the indirect block. In this, the first pointer is dereferenced, leading to a data block. From the data block, byte 4 is taken. 3 blocks are read.

b. immediate data: 2040 B
direct block data: 2^{20} B
indirect block data: $\frac{4096}{4} \cdot 4096$ B,
$= 2^{22}$ B.
These total to $5 \cdot 2^{20} + 2040$ B.

c. i The time of creation is stored in the file's control information because if a file is moved, it should keep its original creation time.

ii The file name is used only in referring to a file, and a file can be referred to from many different places, so the name is stored in the directory.

iii File access rights should be decided by the creator of the file, not the users referring to it. Hence, file access rights are stored with the file.

d. For reading, this structure is very quick because, after the initial read of the control information, we know exactly which block to address based on the position of the required byte. Also, because the blocks of the file tend to be close together, the read arm of the hard disk doesn't need to move much.

Writing is similarly fast as long as there are enough free data blocks in the file's current location. Otherwise, when the file expands, it will have to be moved to a different area of the disk. This can be a very costly operation, and also makes the system liable to external fragmentation.

If the disk has accumulated external fragmentation, it may be difficult to find a place for a new file. In this case, the disk will need to be defragmented to accomodate the new file, which is a very costly operation.

8 a i In demand paging, pages are only ~~assigned to processes~~ created and assigned to physical memory when a process needs more memory. A demand paging system keeps a page table containing the currently assigned pages and their physical addresses. When a new page is added, a physical address not found in the page table must be assigned to it, then it is added to the page table.

ii Temporal locality of reference describes how long it was since a value was last required.

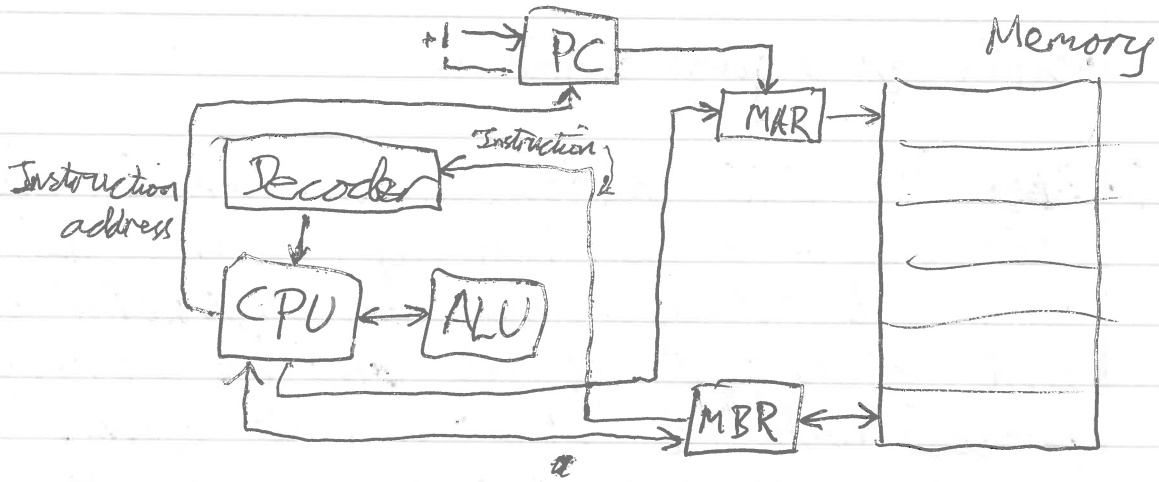
iii Since recently ~~used~~ used values are likely to be used again soon, page replacement should avoid invalidating recently-used pages.

iv Spatial locality of reference describes how close recently-used physical addresses have been to the address in question.

v. New pages should be allocated to areas of physical memory that have had much recent activity, because it is quicker to access close together pieces of memory due to ~~cache~~ caching opportunities.

b i

9 a



Using the address stored in the program counter, an instruction is read from memory. After this, the program counter is incremented and the instruction is passed along to the decoder. This passes the instruction (or part of it) onto the relevant piece of logic in the CPU, which executes the instruction.

Arithmetic instructions are handled by the ALU. Binary operators (like addition) are usually handled by loading one number into a register from memory, then passing the other number as an argument with the instruction. It is also possible to have two numbers loaded from memory, then the binary addition instruction being passed in afterwards.

Memory access works via load and store instructions, with the memory address being an operand or already loaded into a register. The address is passed to the memory address register, and the memory buffer register is set to read or write. Data comes and goes through the MBR.

Control flow instructions are handled by overwriting the value in the program counter.

usually via a jump command.

9 i An I/O device is accessed in either a blocking or a non-blocking manner. If accessed in a blocking manner, an instruction will be sent to the I/O device and the running process will be put in the wait queue. When the device signals that it has finished, the process will be moved from the wait queue to the ready queue, and will eventually start running again. With non-blocking I/O, an instruction is sent through the kernel to the device, which should immediately yield a value, which is brought back to a register for use by the program running.

ii If running in user mode, the program has to request temporary elevation to supervisor mode to access certain instructions. Alternatively, some I/O actions (like reading an owned file with read permissions) are implemented as instructions that don't require use of supervisor mode.

10 a The von Neumann architecture allows programs to be stored as data, allowing the computer to read and write its own programs.

b For values stored by programs that are not part of the OS, the programs storing those values keep track of whether the values are signed by the instructions they use. However, the OS doesn't store anywhere whether the values

themselves are signed.

c A context switch occurs when a job is swapped in in place of the currently running job. In this the new job's virtual memory address assignments must be found and swapped with the current job's assignments. Register values also need to be swapped. Additionally, if a TLB is being ~~se~~ used, it has to be invalidated.

11 a. i. 38144

ii. -5376

iii. -27392

iv. ~~X6XXXX~~

Assuming the number is broken up as follows:

1 00101 0100000000

sign exponent mantissa

in big-endian order,

$N = -2^{5-15} \times 1.01_2$

$= -5 \times 2^{-12}$

$= -0.001220703125$

b. 1001 0101 0000 0000 +

1001 0101 0000 0000

0010 1010 0000 0000

C = 1 because the result is too large, as evidenced by the '1' in the 2^{16} column of the result.

V = 1 because two numbers that would be interpreted as negative in ~~to~~ two's complement

arithmetic seemed to give a result representing a positive number.