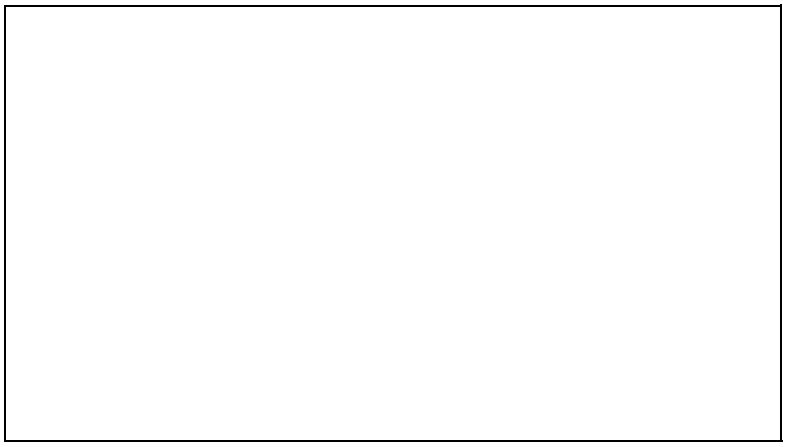
PERMANENT STORAGE



5:29 / 5:29 1.0x

Now we'll look at how we incorporate permanent storage into our computer

system with the other components that we've already seen.

So first of all, we have one or more processors.

And each of those processors has caches.

And those connect typically to a second level of cache, an L2 cache.

And that connects then to memory.

Now, we need permanent storage because that's where our apps lie.

That's where we have our code and our data associated with the different apps

that we run.

And we connect that also to main memory.

So as we load an application from permanent storage,

we bring it into memory.

And we bring in the code, and we bring in some of its associated data.

Now, we also do have a connection from the processor

that we're not showing here.

So the processor-- because this is input/output,

as we've talked about previously-- will have some registers

associated with permanent storage that our processor can see also

and can control.

Now, permanent storage is an input/output device.

Help

It does both input and output.

So input is when we're transferring from storage into memory.

And output is when we're transferring from memory into storage.

So for instance, when we load our app we're doing input,

and when we're updating data associated with our app then we're doing output.

The permanent storage in our phone uses flash memory.

This is a type of memory that permanently stores

our apps and the associated data.

So it's non-volatile.

That means when you turn off the power, you don't lose the data.

And this is different than the DRAM and SRAM memories

that we've seen previously.

When you turn off the power, you lose the data in those types of memories.

Now, when an app opens, a portion of its code and data

is transferred for permanent storage into memory,

where it's accessed by the processor.

Now, we don't bring in the entire code and the entire data.

We just bring what we need right now.

And then additional code and data is brought into memory on demand.

So when we need to bring in more of our program,

when we access that part of the code, at that point

we transfer that code into memory from permanent storage.

Now, flash is non-volatile and relatively cheap and dense,

but unfortunately it's very slow.

A read from flash memory may take about 25 microseconds.

Now, this is 50,000 processor clock cycles,

if your processor runs at two gigahertz, which is a long, long time.

And writes can be 10 times slower than reads.

So if you're updating data in your app, it may be much slower even then.

Help

Now, how do we prevent the processor from stalling

when it takes 50,000 cycles to transfer data from flash into memory?

Well, what we do is we'll switch to another task

while waiting for the access.

Remember when we talked about multithreading

and we talked about having multiple programs that

are ready to run in our phones?

Well, if one program has to go to flash memory,

then we can switch to a different program

while we're waiting for that transfer to happen.

Now, we're also going to take advantage of spatial locality,

just as we did with caches.

So we're going to transfer large chunks of data between permanent storage

and main memory.

And data here means instructions or actual data.

So we're not going to just get one instruction or two,

we're going to get many, many instructions when we request them

from permanent storage.

So this is similar to transferring blocks between cache and main memory.

Now, in addition to transferring blocks of 32 to 128 bytes

typically between memory and the caches, now we're going to transfer pages.

Now, because reading from permanent storage

is so slow, then when we go there we're going to get a lot of data

or a lot of instructions.

So our pages are much bigger than our cache blocks.

So those are 4 to 16 kilobytes in size typically.

And this depends on the operating system. When a cache block is not

found in either cache or memory,

now we're going to transfer a page, this larger

Help

4 to 16 kilobyte chunk of instructions or data, from storage to memory.

Now, the operating system gets involved here,

because it determines which page to replace in memory-- assuming memory

is full-- and then what other tasks to run in the processor

while this first task has its data transferred from storage to memory.

Now, some special hardware called Direct Memory Access, or DMA hardware,

gets involved, because it's the thing that

performs the transfer while the processor runs the other task.

So the operating system doesn't have to do this transfer by itself.

It sets up the DMA hardware and tells it what transfer it wants it to make.

Download transcript .txt

Show Discussion

New Post

1. CHECK YOUR UNDERSTANDING (1/1 point)

Which of the following describe approaches for reducing performance loss due to the slowness of Flash storage? **[Check all that apply]**

- [x] Transferring pages of instructions or data between Flash storage and main memory. ✔
- [x] Switching to another task during transfers between Flash storage and main memory. ✔
- [x] Using dedicated Direct Memory Access (DMA) hardware to perform transfers between Flash storage and main memory. ✔

EXPLANATION

By transferring a page that is typically kilobytes in size, we exploit spatial locality and reduce how often we need to access Flash storage.

By switching to another task, we can perform useful work on the processor in parallel with the transfer (another example of parallelism).

By using dedicated DMA hardware, the processor can be used to run a different task.

Final Check | **Save** | Hide Answer | *You have used 1 of 2 submissions*

Help

Show Discussion

New Post