


Exercise Session 4

Computer Systems



Totally new and
original slides!

Agenda

- Scheduling
 - Terminology
 - Round Robin
 - Shortest Job First
 - Earliest Deadline First
 - Priority Inversion
- I/O
 - Interrupts
 - DMA

Terminology

- Waiting time or turnaround time
 - The time take to finish the job from the point where it entered the system
- Hold time
 - The time taken to start executing the job from the point where it arrives
- Response time
 - The time taken to respond to a request for service

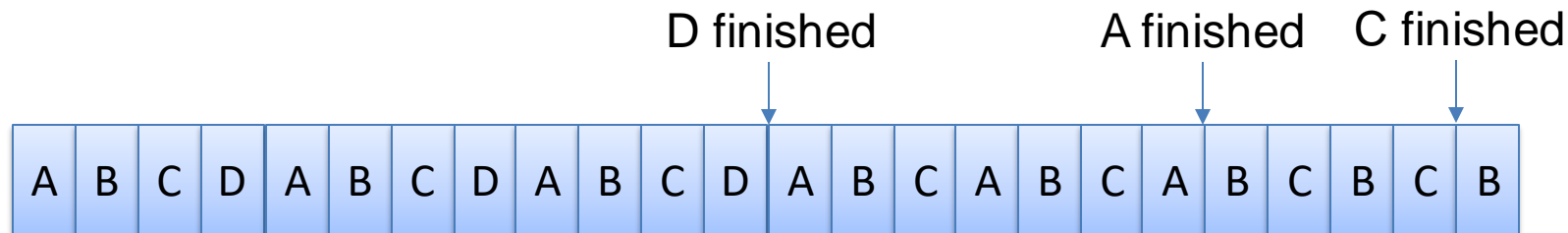
Terminology: Preemptive Scheduling

- Processes dispatched and descheduled without warning
 - Often on a timer interrupt, page fault, etc.
- The most common case in most OSes
- Soft realtime systems are usually preemptive
- Hard realtime systems are often not!

Round Robin

- Run processes one after another in a circular fashion.
- Example 1 ms slices

Process	Execution time (ms)
A	6
B	8
C	7
D	3



Shortest-Job First

- Always run process with the shortest execution time.
- Optimal: minimizes waiting time (and hence turnaround time)

Process	Execution time
A	6
B	8
C	7
D	3



Execution time estimation

- Problem: what is the execution time?
 - For mainframes, could punt to user
 - And charge them more if they were wrong
- For non-batch workloads, use CPU burst times
 - Keep exponential average of prior bursts
- Or just use application information
 - Web pages: size of web page

SJF & preemption

- Problem: jobs arrive all the time
- “Shortest remaining time next”
 - New, short jobs may preempt longer jobs already running
- Still not an ideal match for dynamic, unpredictable workloads
 - In particular, interactive ones.

Earliest Deadline First

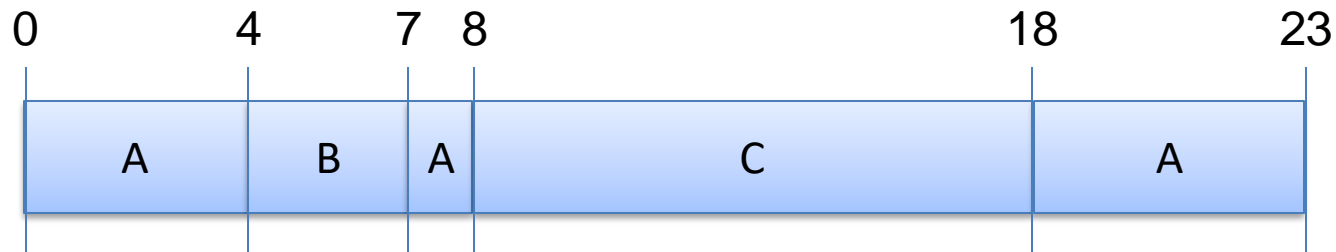
- Schedule task with earliest deadline first (duh..)
 - Dynamic, online.
 - Tasks don't *actually* have to be periodic...
- EDF will find a feasible schedule if:

$$\sum_{i=1}^m \frac{C_i}{P_i} \leq 1$$

- What does this mean? If EDF can't do it, nobody can!
- Which is very handy. Assuming zero context switch time...

Earliest Deadline First

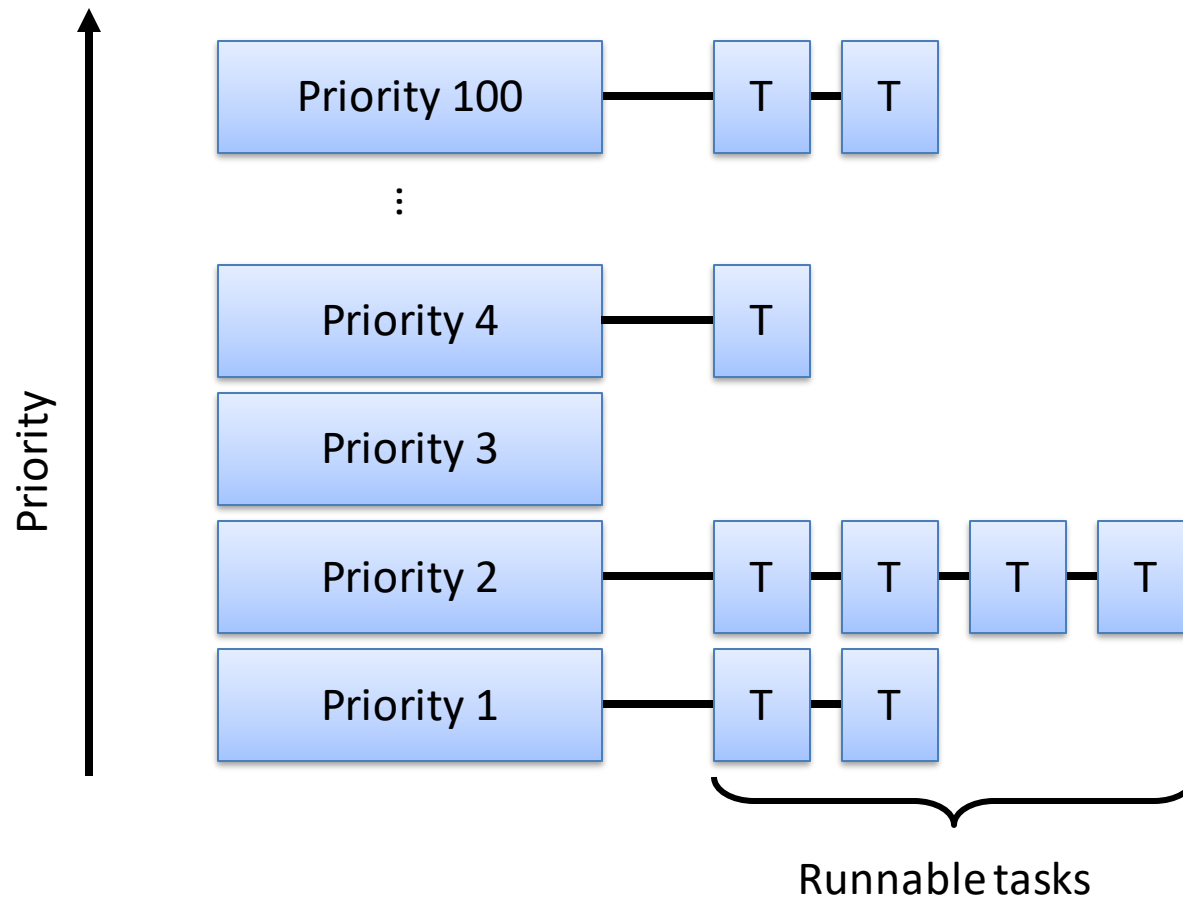
Process	Arrival	Duration	Deadline
A	0	10	25
B	4	3	8
C	8	10	18



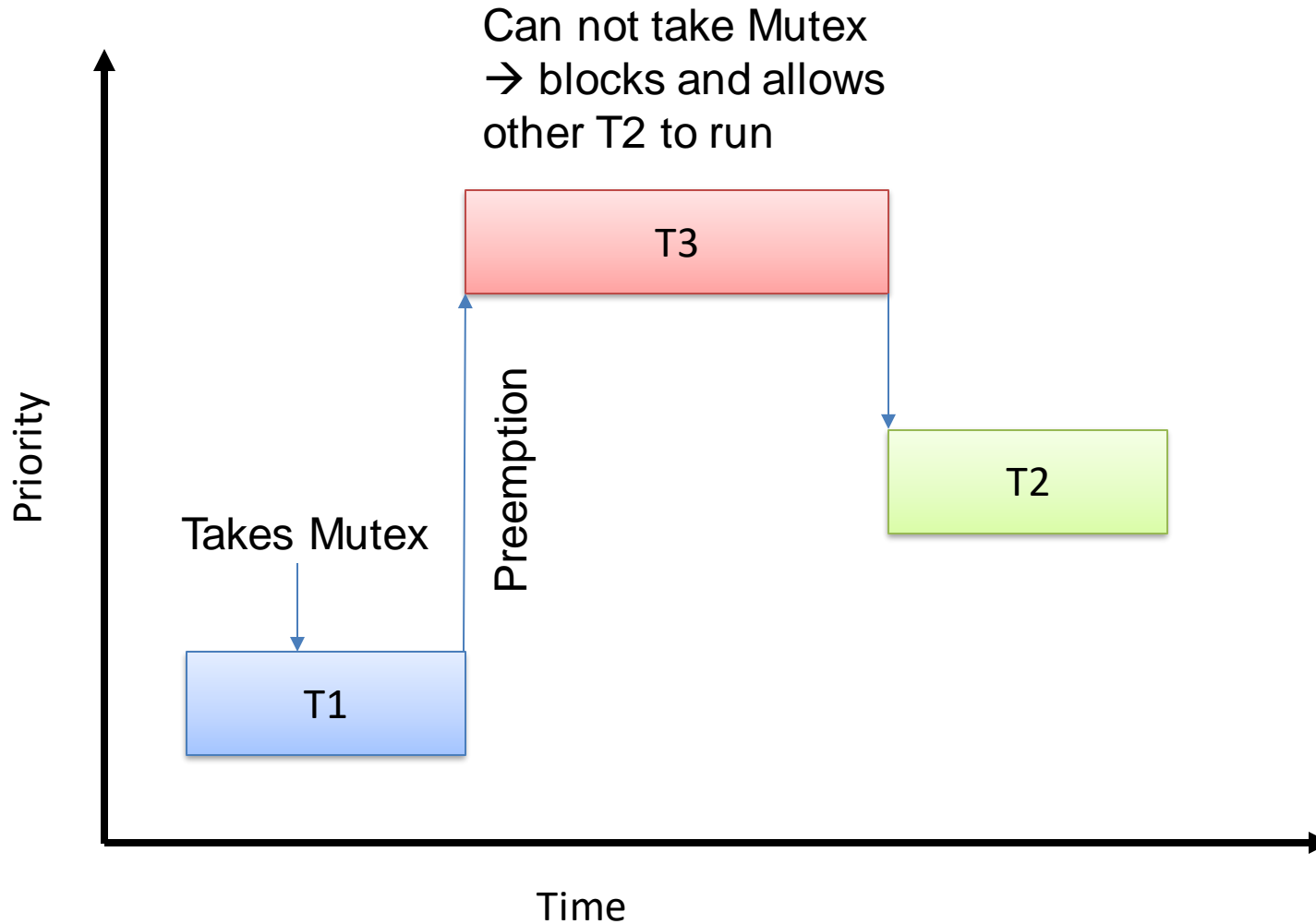
Priority

- Very general class of scheduling algorithms
- Assign every task a priority
- Dispatch highest priority runnable task
- Priorities can be dynamically changed
- Schedule processes with same priority using
 - Round Robin
 - FCFS
 - Etc.

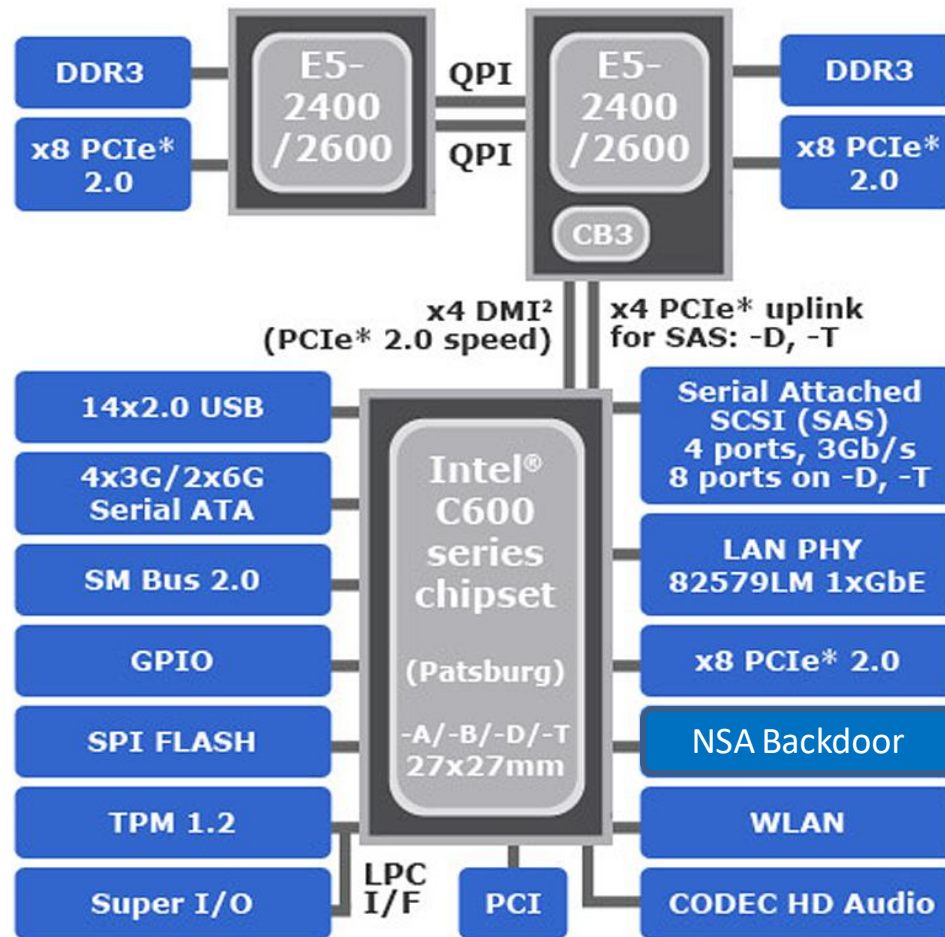
Priority queues



Priority Inversion

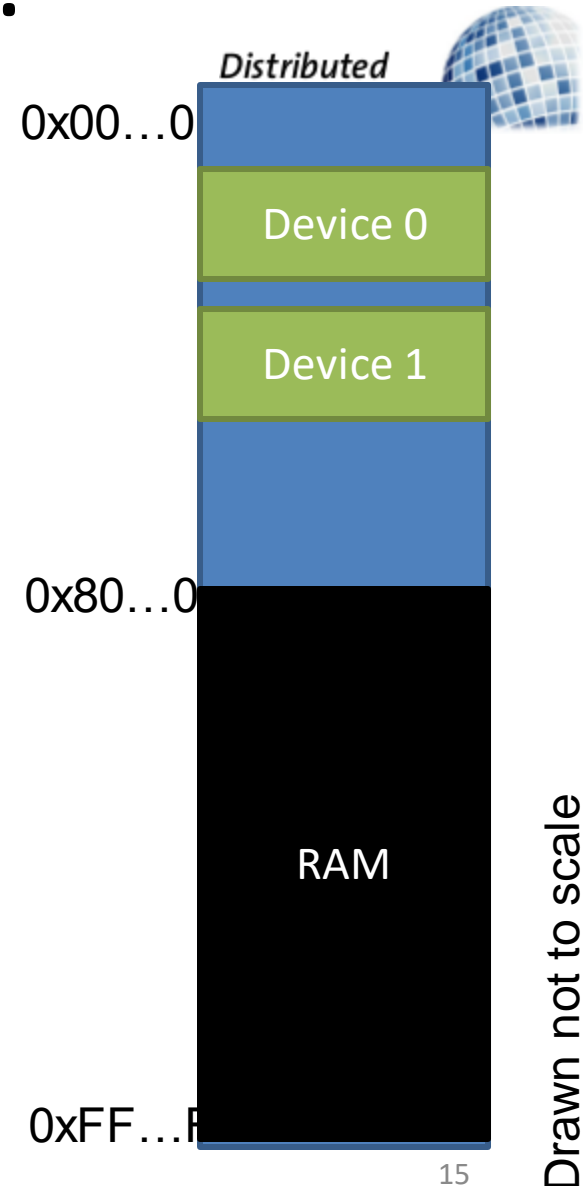


Devices



How to access devices?

- Memory mapped
 - Write & Read to devices as you have done with normal memory (movl...)
 - Or through special address space (mostly legacy devices)
 - Within a device's address space, there are many registers





MANY Registers...

20.17 Register summary – USB module

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	ENABLE	SPEED	FIFOEN	STFRNUM	MAXEP[3:0]				238
+0x01	CTRLB	–	–	–	PULLRST	–	RWAKEUP	GNACK	ATTACH	238
+0x02	STATUS	–	–	–	–	URESUME	RESUME	SUSPEND	BUSRST	239
+0x03	ADDR	–	ADDR[6:0]							239
+0x04	FIFOWP	–	–	–	FIFOWP[4:0]					239
+0x05	FIFORP	–	–	–	FIFORP[4:0]					240
+0x06	EPPTRL	EPPTR[7:0]								240
+0x07	EPPTRH	EPPTR[15:8]								241
+0x08	INTCTRLA	SOFIE	BUSEVIE	BUSERRIE	STALLIE	–	–	INTLVL[1:0]		241
+0x09	INTCTRLB	–	–	–	–	–	–	TRNIE	SETUPIE	242
+0x0A	INFLAGSACLR	SOFIF	SUSPENDIF	RESUMEIF	RSTIF	CRCIF	UNFIF	OVFIF	STALLIF	242
+0x0B	INFLAGSASET	SOFIF	SUSPENDIF	RESUMEIF	RSTIF	CRCIF	UNFIF	OVFIF	STALLIF	242
+0x0C	INFLAGSBCLR	–	–	–	–	–	–	TRNIF	SETUPIF	243
+0x0D	INFLAGSBSET	–	–	–	–	–	–	TRNIF	SETUPIF	243
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	Reserved	–	–	–	–	–	–	–	–	
+0x10-0x39	Reserved	–	–	–	–	–	–	–	–	
+0x3A	CAL0	CAL[7:0]								243
+0x3B	CAL1	CAL[15:8]								244

Simple Microcontroller USB Interface

Devices are **NOT** memory

- Contents of register may change unexpectedly
 - Data received
- Writing to a register trigger actions
 - Shutdown device / machine
 - Perform reset

IMPORTANT



Devices

- Writing device drivers is tedious
 - Setting a single bit wrong and the device does not work
 - Debugging is hard: Likely to set a bit wrong due to a wrong shift & mask.



Devices & Caches

- Device registers cannot be cached due to inconsistency problem i.e. register content changes w/o CPU write!
- What about cache lines? Would overwrite other register values when writing back
- Set the “no-cache” flag in the page table entry

Interrupts

```
1. #define UART_BASE 0x12345000
2. #define UART_THR (UART_BASE + 0)
3. #define UART_RBR (UART_BASE + 4)
4. #define UART_LSR (UART_BASE + 8)

5. void serial_putc(char c)
6. {
7.     volatile char *lsr = (char *) UART_LSR;
8.     volatile char *thr = (char *) UART_THR;

9.     // Wait until FIFO can hold more chars
10.    while((*lsr & 0x20) == 0);
11.
12.    *thr = c
13. }
```

- Any “problems” with that one?

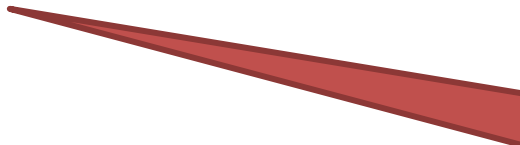
Interrupts

```
1. #define UART_BASE 0x12345000
2. #define UART_THR (UART_BASE + 0)
3. #define UART_RBR (UART_BASE + 4)
4. #define UART_LSR (UART_BASE + 8)

5. void serial_putc(char c)
6. {
7.     volatile char *lsr = (char *) UART_LSR;
8.     volatile char *thr = (char *) UART_THR;

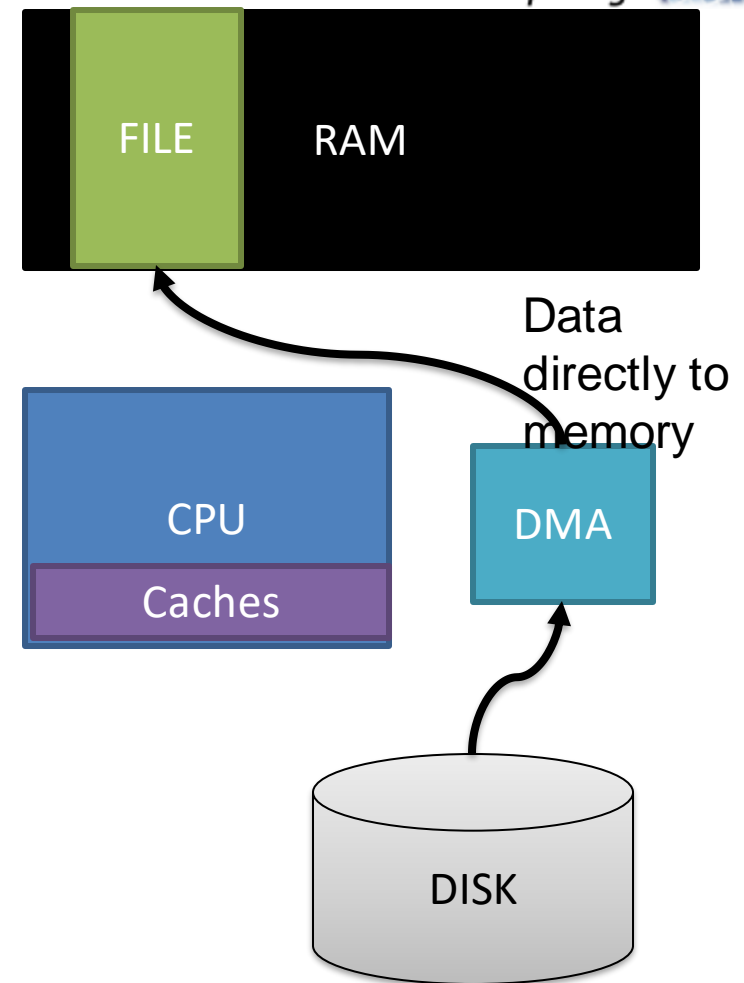
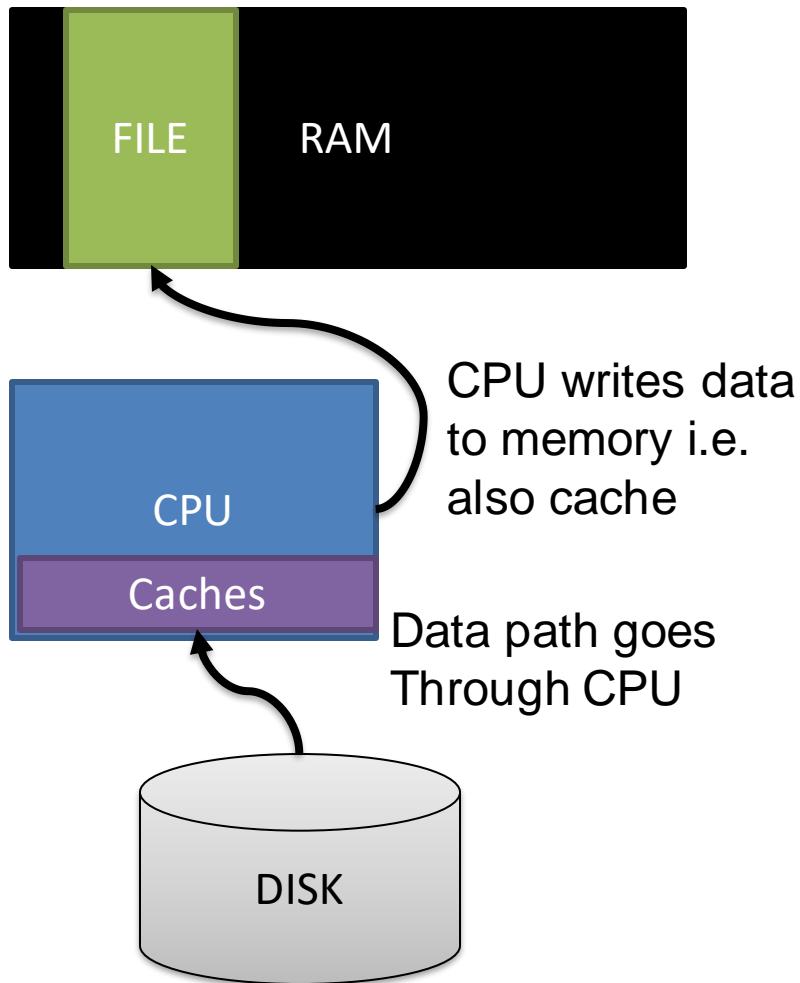
9.     // Wait until FIFO can hold more chars
10.    while((*lsr & 0x20) == 0);
11.
12.    *thr = c
13. }
```

- Register callback for an interrupt
- Activate interrupt

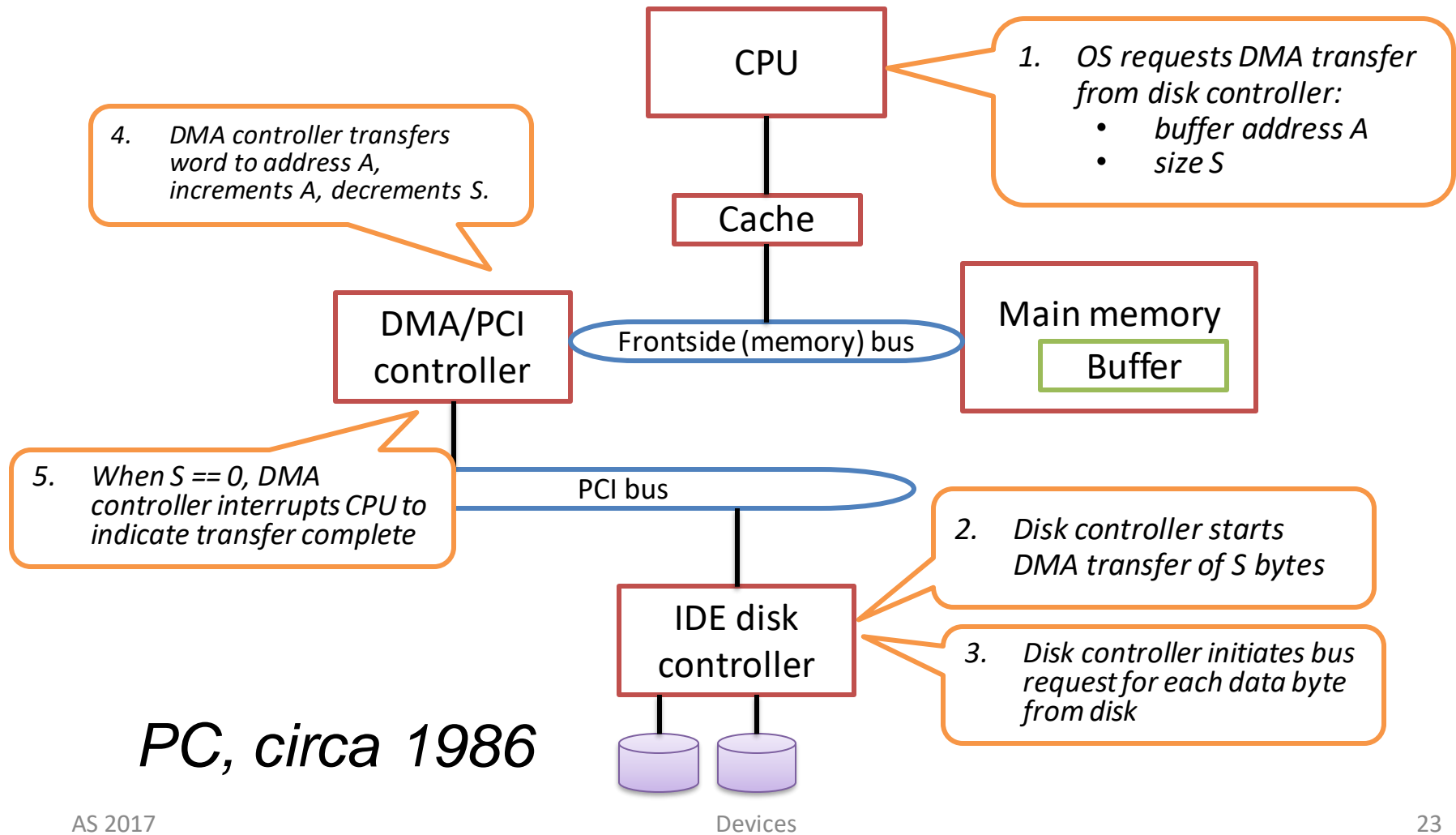


Polls register
=
wastes CPU cycles

DMA



Very simple DMA transfer

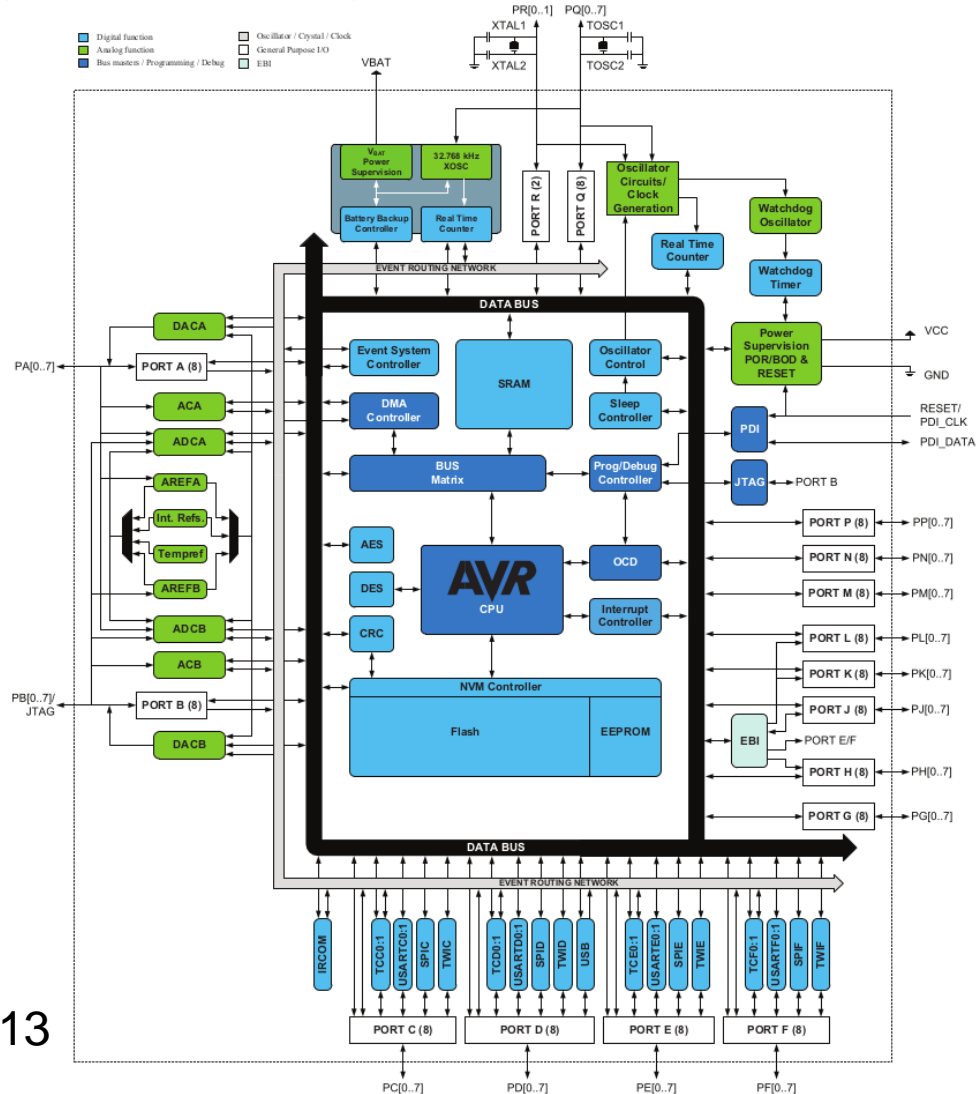


PC, circa 1986

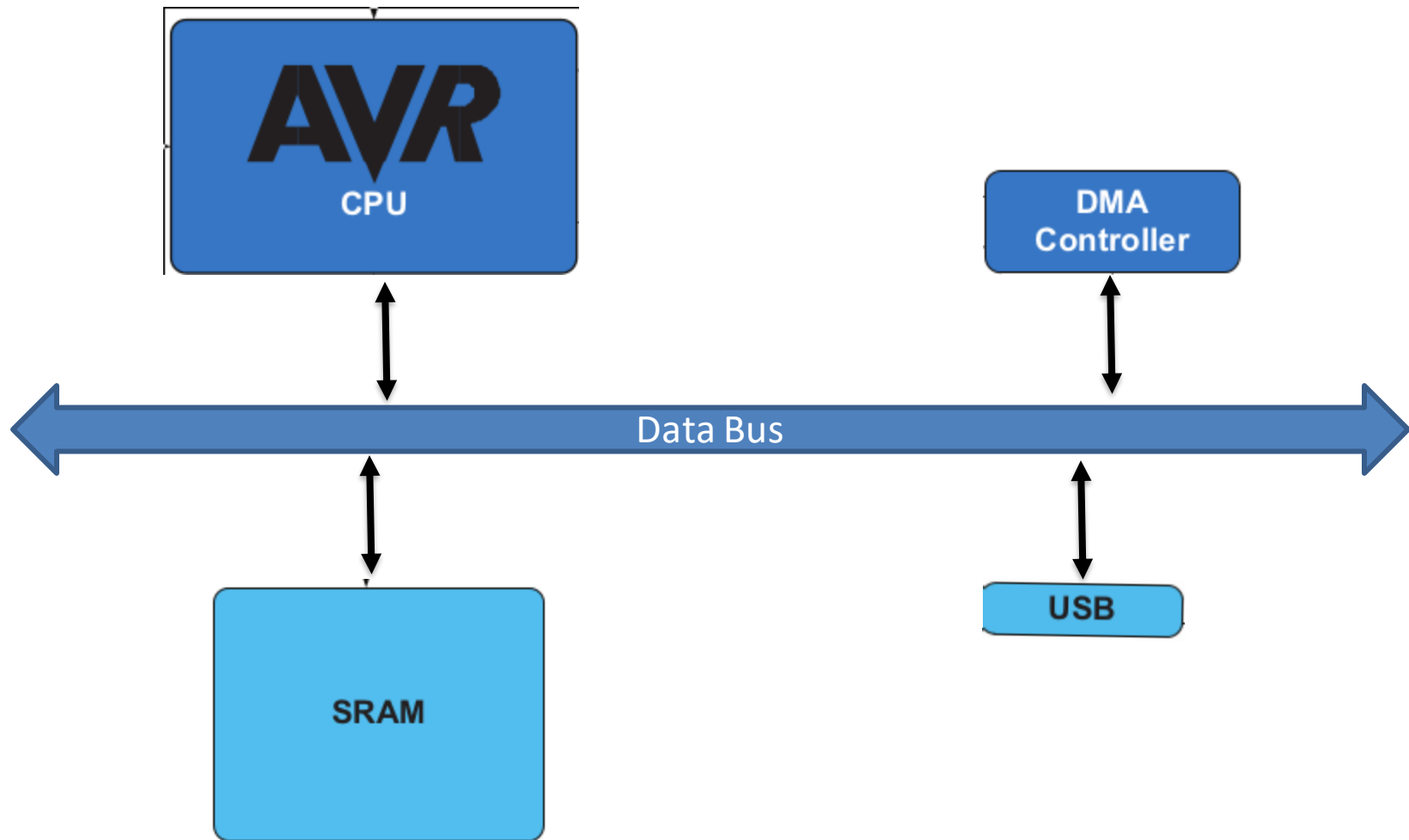
Very simple DMA transfer



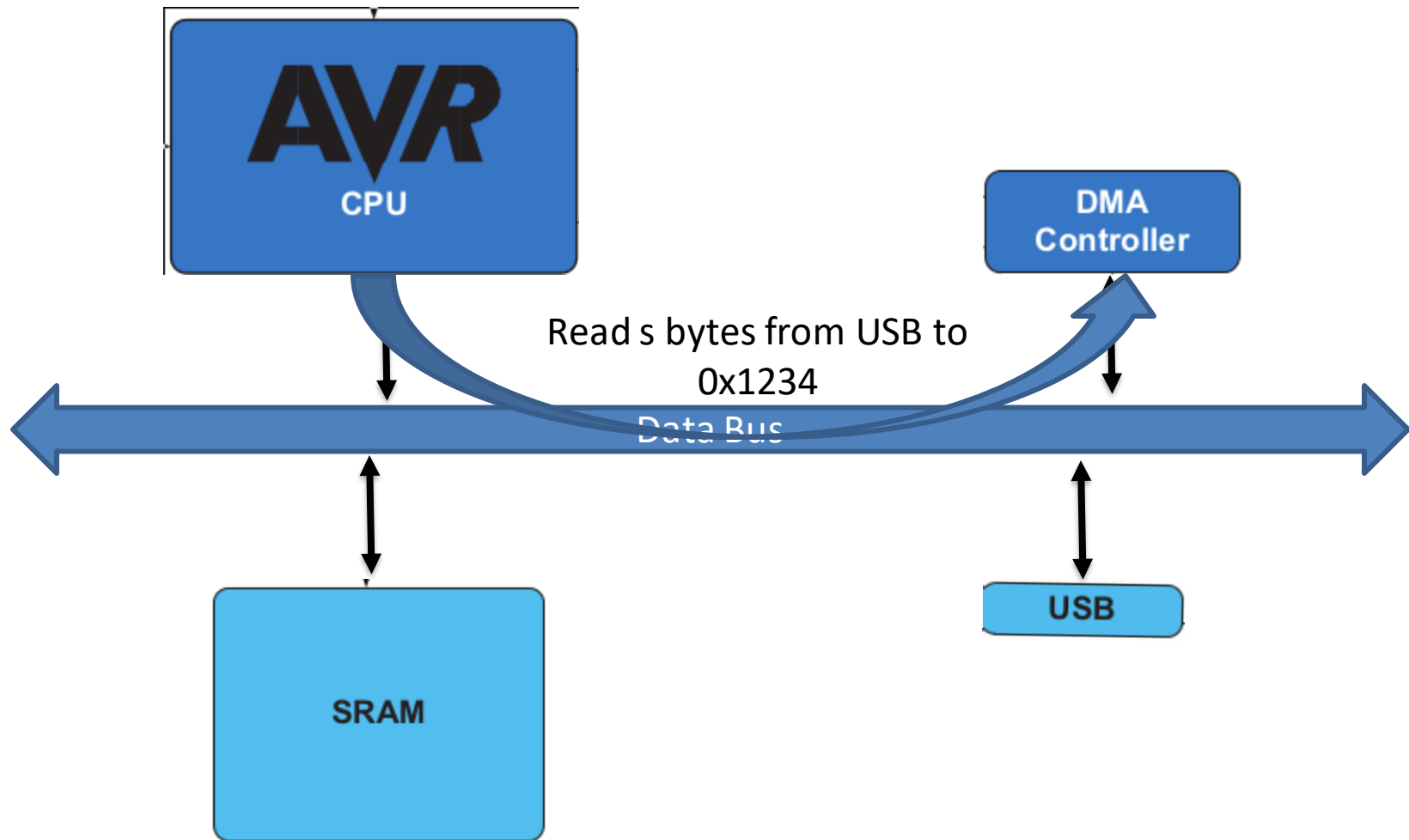
Figure 2-1. Atmel AVR XMEGA AU block diagram.



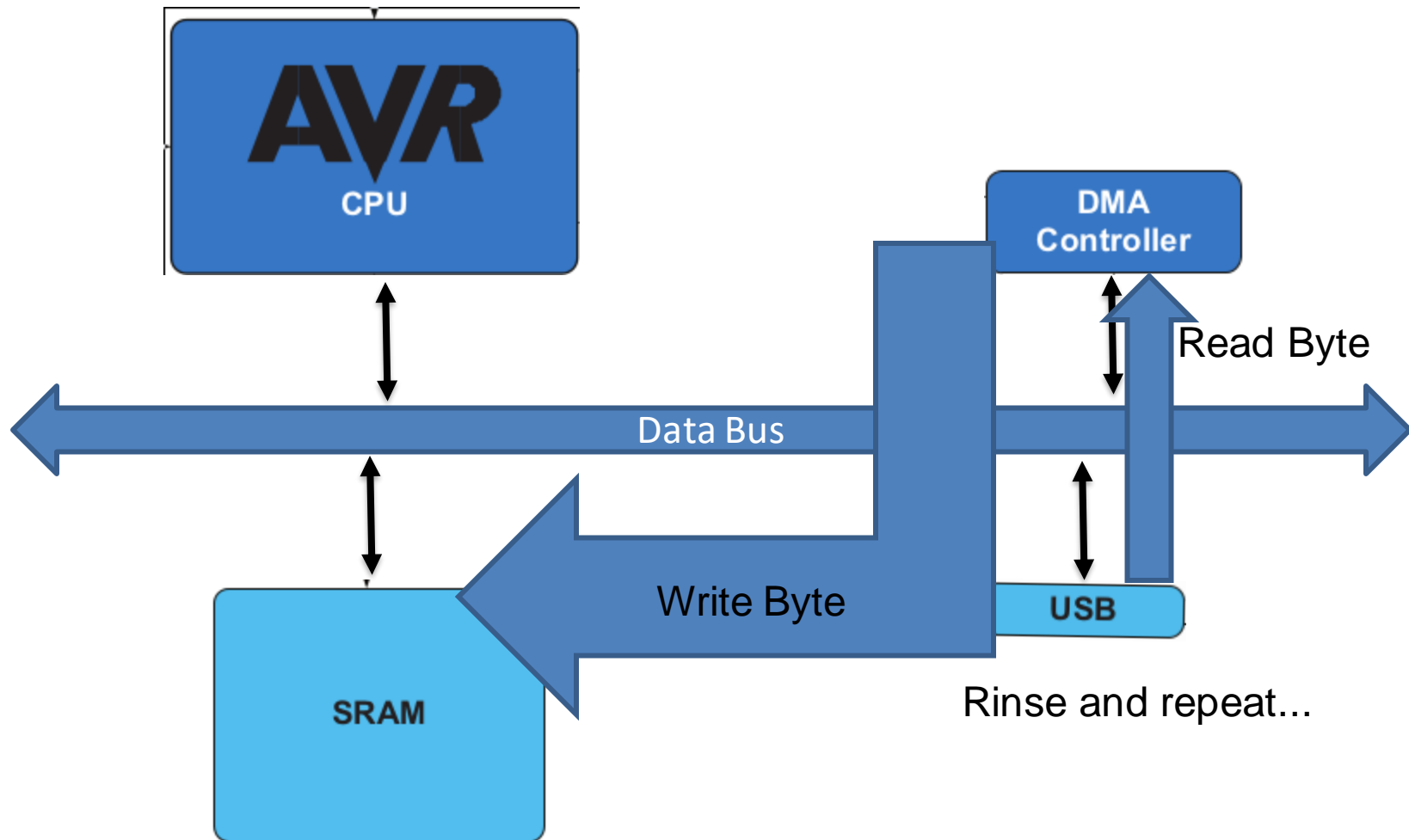
I said simple...



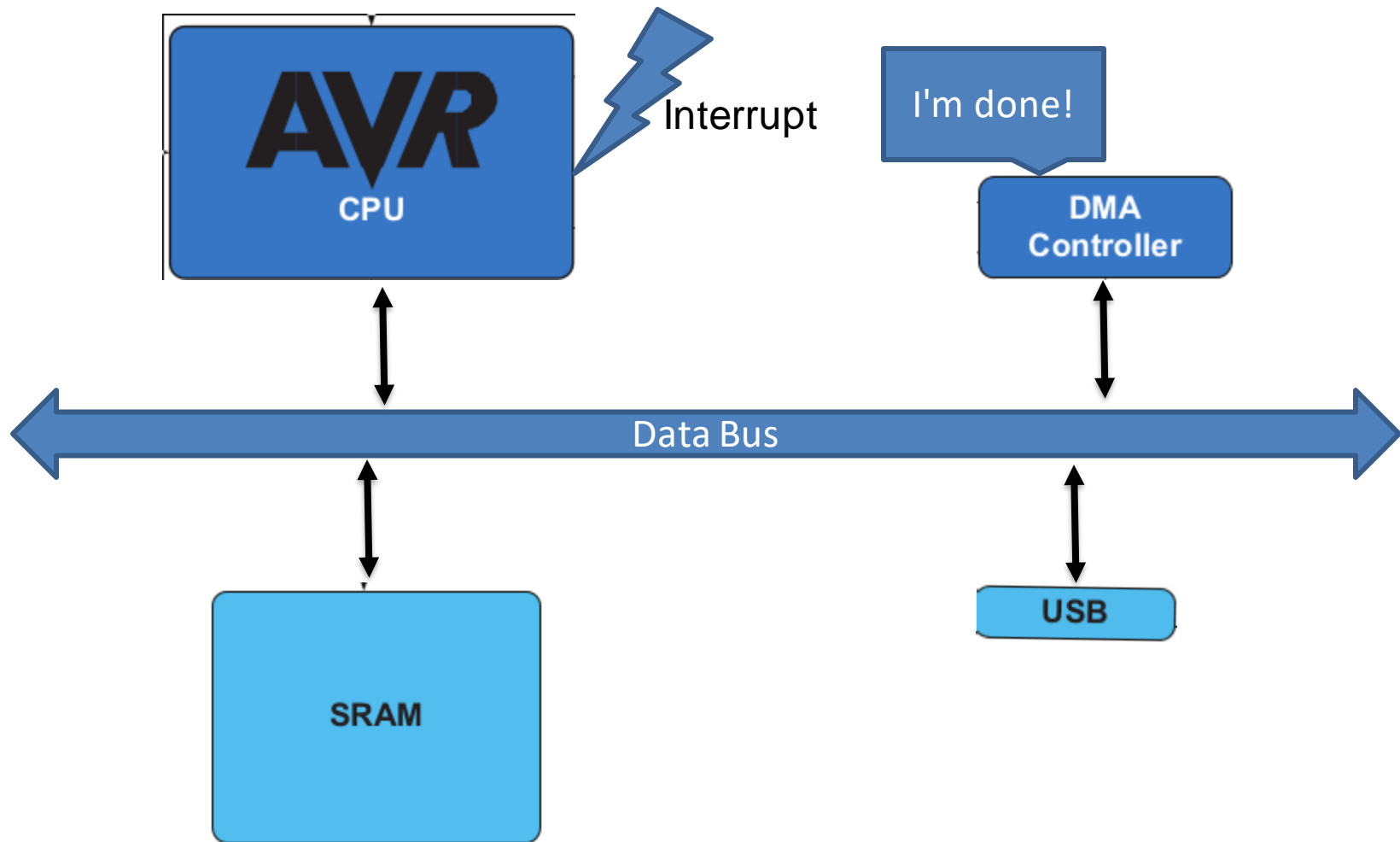
I said simple...



I said simple...

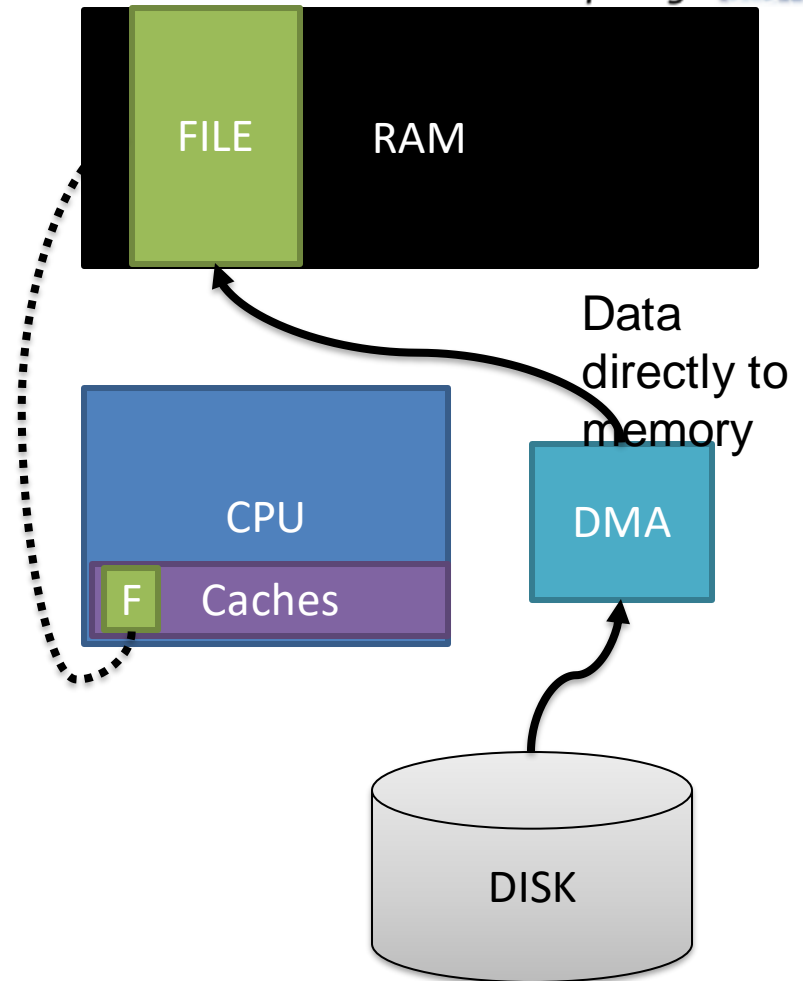


I said simple...



DMA

- DMA is like device registers! Changes the contents w/o CPU write.
- Cannot “no-cache”:
Bad performance
- Explicitly invalidate cache!



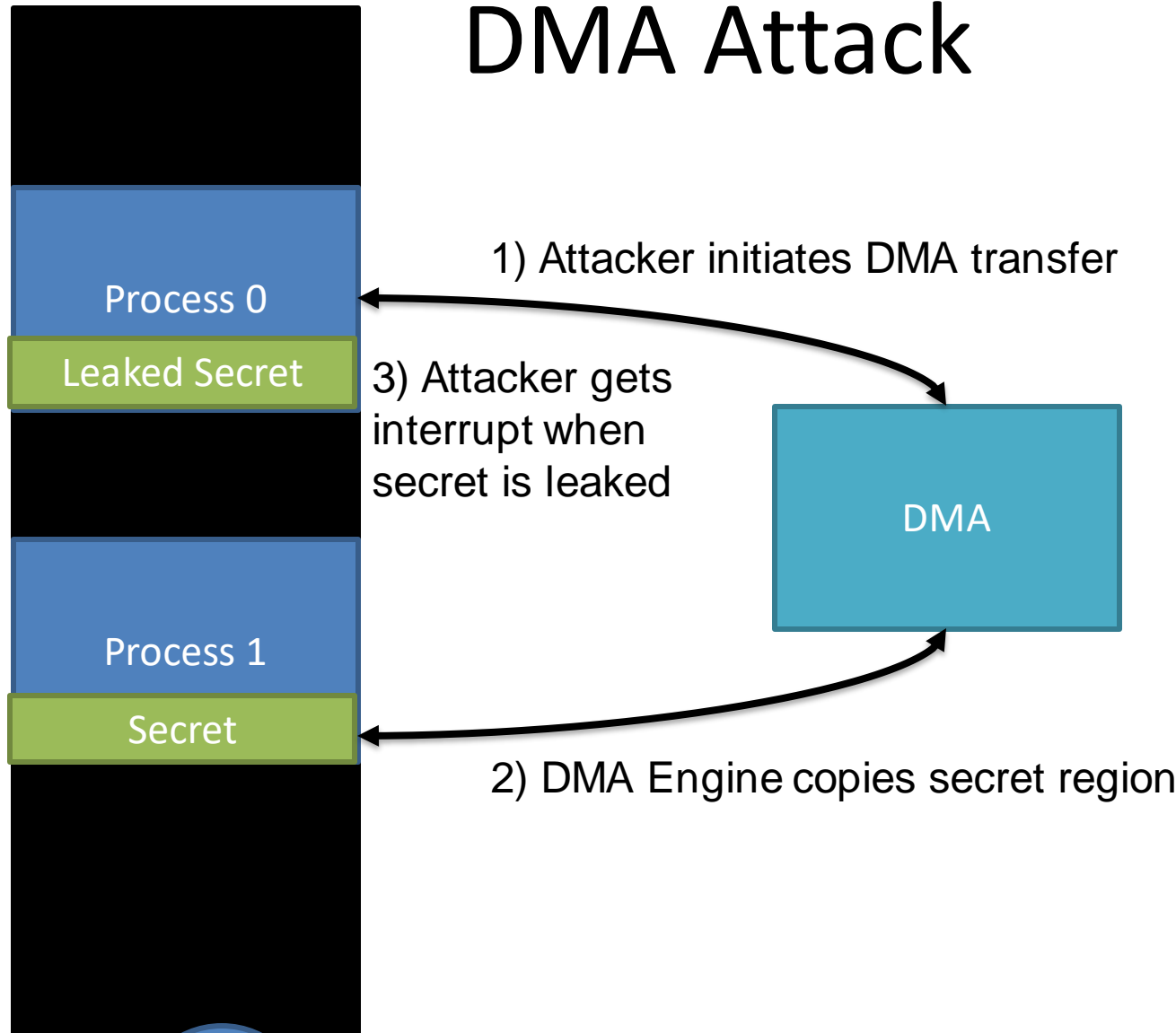
DMA

- Deal with physical addresses only!
 - Any problems with that?

DMA

- Deal with physical addresses only!
 - Any problems with that?
 - Programs deal with virtual addresses: need translation!
 - Physical Range may not be contiguous (gather/scatter)
- How about security?
 - Unrelated aside: Anyone remember FireWire?

DMA Attack



Anyone remember FireWire?

http://en.wikipedia.org/wiki/DMA_attack

I/O Protection

I/O operations can be dangerous to normal system operation!

- Dedicated I/O instructions usually privileged
- I/O performed via system calls
 - Register locations must be protected
- DMA transfers must be carefully checked
 - IOMMU