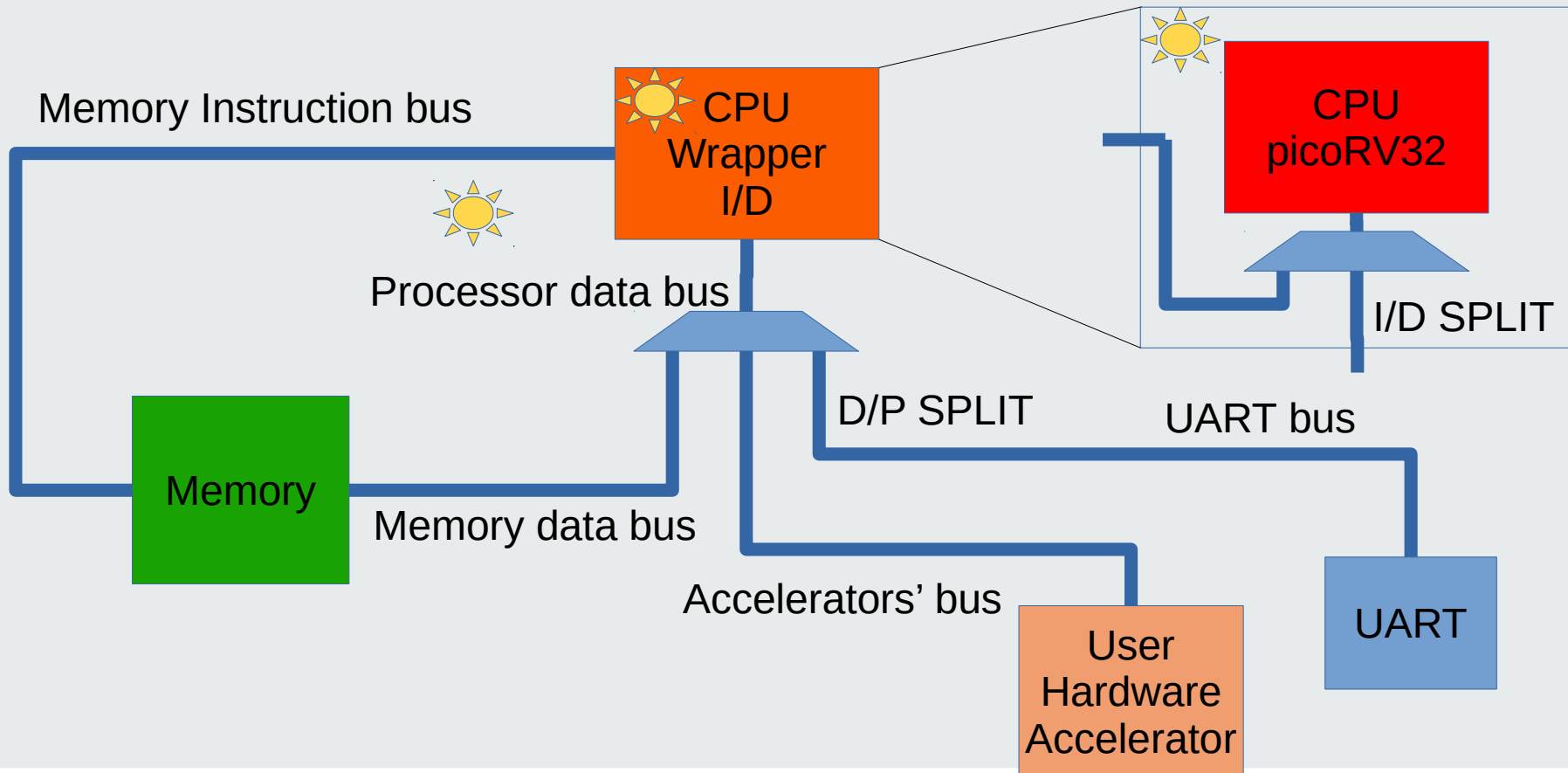


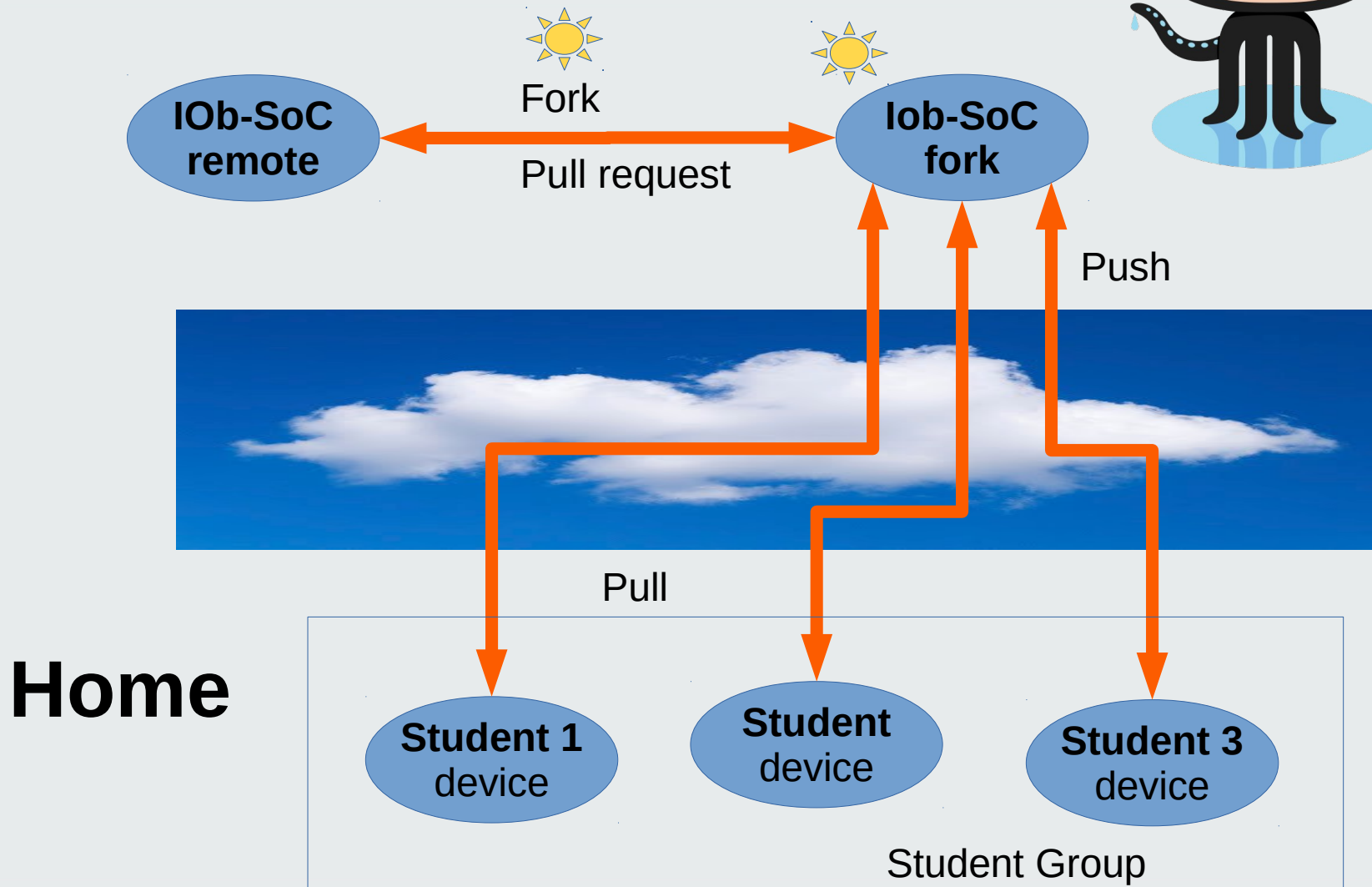
Computer Electronics

Lecture 3: Iob-SoC Tour

lob-SoC: recap of last lesson & more



Github



IOb-SoC: quick recap

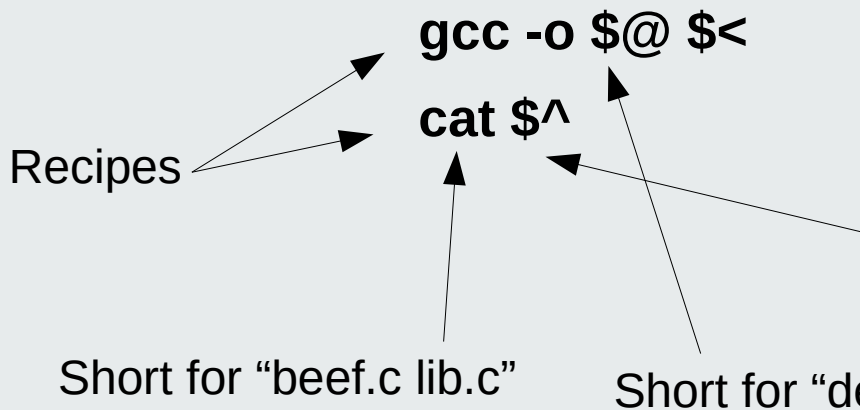
- The PicoRV32 RISC-V CPU (open-source)
- Boot ROM (has small program to load user programs)
- SRAM (on-chip memory for running small programs or storing fast access data)
- UART interface for loading the program and for communication
- *Instruction and Data L1 Caches + shared L2 cache*
- *Other optional modules: Ethernet, UART, DMA*
- **Ready to attach user modules**

Make recap, the include directive



include "my_wonderful_makefile_segment.mk"

Target → **dead:** **beef.c lib.c** ← *Prerequisites*



Short for "beef.c"

IOb-SoC Tour

- System configuration
- Firmware
- 32-bit CPU memory organization
- SRAM
- Peripherals
- Boot sequence
- Example boot sequence upgrade to use flash memory
- Top Makefile

IOb-SoC: system configuration

- Accomplished using makefile segment **system.mk**
 - Located at *repo* root
- Parameters of most interest in system.mk
 - `FIRM_ADDR_W` = \log_2 (firmware size in bytes)
 - `SRAM_ADDR_W` = \log_2 (SRAM size in bytes)
 - `PERIPHERALS` = list of peripherals

system.mk: FIRM_ADDR_W

- $\text{FIRM_ADDR_W} = \log_2(\text{firmware size in bytes})$
- Note the `_W` suffix: stands for WIDTH in bits
- The firmware bytes can be addressed with `FIRM_ADDR_W` bits
- Then there are $2^{\text{FIRM_ADDR_W}}$ bytes in the firmware



Firmware

- Firmware is the resident software of an embedded system
- Sits on non-volatile memory: ROM, EPROM, Flash
- It may never change...
- Some more complex devices may allow firmware upgrades (eg. smartphone)

Memory organization for a 32-bit CPU

	0x00	Byte 3	Byte 2	Byte 1	Byte 0
	0x04			...	Byte 4
	0x08				
Full word addresses = Byte address >> 2	0x0C				
Half word addresses = Byte address >> 1	0x10				
	0x14	Byte 23	...		Byte 20

A 32-bit CPU can

- With single instruction can
 - Read and write single bytes (char, uchar)
 - Read and write single half words (short, ushort)
 - Read and write single long words (int, uint)
- With multiple instructions can
 - Read and write multiple bytes (number array, string)
 - Read and write multiple half words (arrays)
 - Read and write multiple long words (arrays)
- Read = Load, Write = Store

system.mk: SRAM_ADDR_W

- The SRAM bytes can be addressed with SRAM_ADDR_W bits
- Then there are $2^{\text{SRAM_ADDR_W}}$ bytes in the memory
- If SRAM_ADDR_W is greater than FIRM_ADDR_W then the firmware **fits** in the SRAM and can run on it

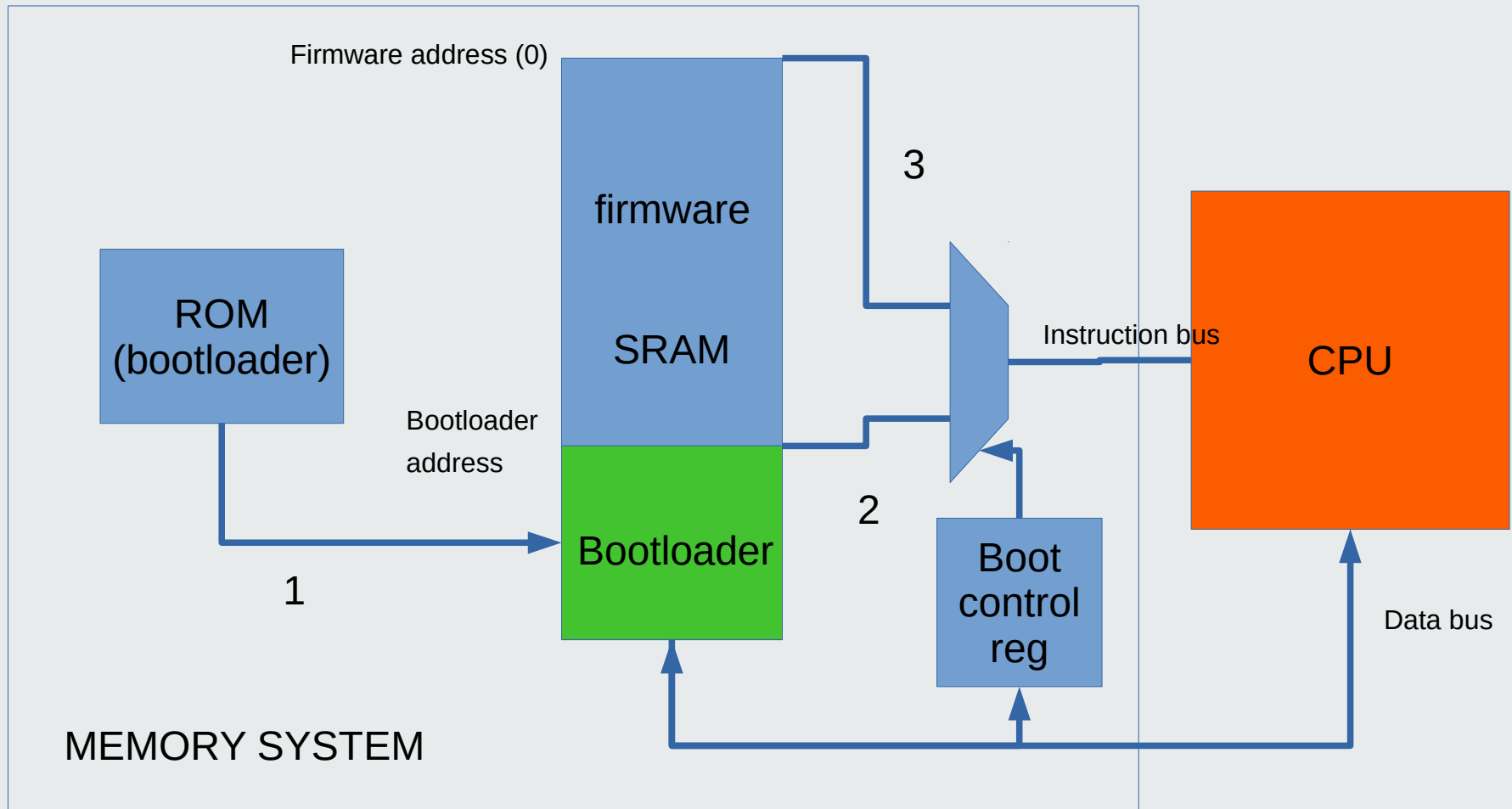
system.mk: PERIPHERALS

- List of peripherals in system.mk
- By default contains only one peripheral: the UART
- Users can make peripherals and attach them to the IOb-SoC peripheral bus
- For this to happen
 - 1) the peripheral repository must follow a certain structure
 - 2) the peripheral repository must be placed in the **submodules** directory
 - 3) the directory name of the peripheral repository must be added to the the PERIPHERALS list

Iob-SoC boot sequence

- 1) Power on; Hardware reset (hard reset)
- 2) Bootloader program copied from ROM to SRAM by a special hardware circuit; bootloader runs on SRAM
- 3) Bootloader awaits for firmware to arrive by UART
- 4) Bootloader receives firmware from UART and loads it to SRAM
- 5) Bootloader sets boot control register and reboots system
- 6) Boot control register tells bootloader to start running from a different address where firmware has been stored

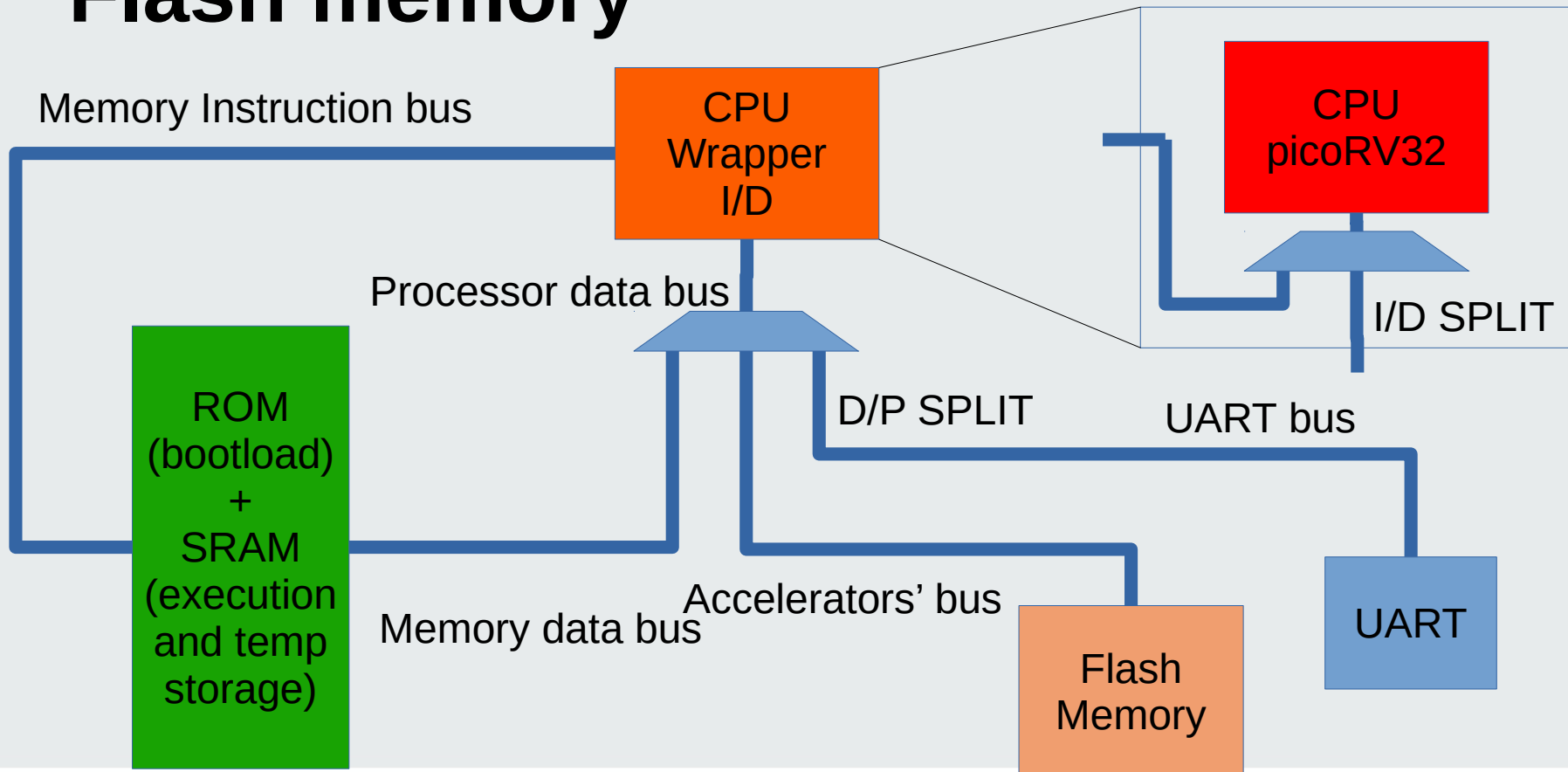
Iob-SoC boot sequence illustrated



Example: Upgrading IOb-SoC with Flash memory

- 1) Power on; Hardware reset (hard reset)
- 2) Bootloader program copied from ROM to SRAM by a special hardware circuit; bootloader runs on SRAM
- 3) Bootloader awaits for firmware to arrive by UART
- 4) Bootloader receives firmware from UART and loads it to FLASH
- 5) Boot loader reboots system (soft reset)
- 6) Boot loader starts again and awaits for firmware to arrive by UART
- 7) Bootloader times out and copies firmware from FLASH to SRAM
- 8) Bootloader sets boot control register and reboots system
- 9) Boot control register tells bootloader to start running from a different address where firmware has been stored

Example: Upgrading IOb-SoC with Flash memory



IOb-SoC: top Makefile

- The top Makefile contains the following *targets*
 - sim: simulate, eg. make sim INIT_MEM=1
 - sim-waves: simulates and displays waveforms using gtkwave
 - sim-clean: cleans all simulation generated files
 - run-sim: cleans sim files, runs simulation and creates a test log
 - firmware: compiles de firmware only
 - bootloader: compiles the bootloader only
 - sw-clean: cleans all software compilation generated files
 - doc: makes all the documents in the document directory
 - doc-clean: cleans all document compilation generated files
 - clean-all: complete detox, cleans every generated files

IOb-SoC: README, LICENSE and test directory

- The README file
 - Several styles exist
 - Adopted style: brief description of the repo followed by what you can do with it – prompts action!
- LICENSE file
 - States under which conditions the files in repo can be used
 - Adopted license created by MIT with 2 clauses only
 - Keep copyright notice in the main files
 - Keep license text in the main files
- test directory
 - Used to place golden test logs in automatic testing
 - Generated test log is compared to golden to yield pass/fail result