

The ARM architecture

Matteo SONZA REORDA
Dip. Automatica e Informatica
Politecnico di Torino



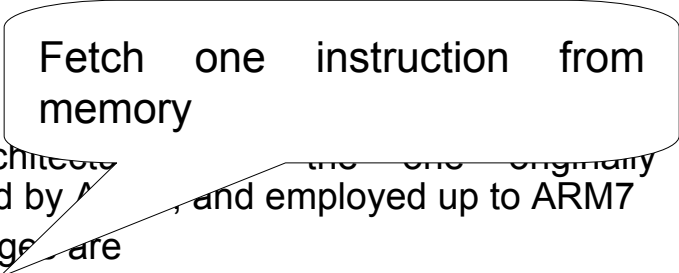
The ARM architecture

- Several ARM processors have been developed and sold.

<u>Core</u>	<u>Architecture</u>
ARM1	v1
ARM2	v2
ARM2aS, ARM3	v2a
ARM6, ARM600, ARM610	v3
ARM7, ARM700, ARM710	v3
ARM7TDMI, ARM710T, ARM720T, ARM740T	v4T
StrongARM, ARM8, ARM810	v4
ARM9TDMI, ARM920T, ARM940T	v4T
ARM9ES	v5TE
ARM10TDMI, ARM1020E	v5TE

3-stage ARM

- This architecture was the one originally developed by Acorn, and employed up to ARM7
- The 3 stages are
 - Fetch
 - Decode
 - Execute
- Some instructions (e.g., those accessing the memory) require more than 3 clock cycles to be executed



Fetch one instruction from memory

- This architecture was the one originally developed by Acorn, and employed up to ARM7
- The 3 stages are
 - Fetch
 - Decode
 - Execute
- Some instructions (e.g., those accessing the memory) require more than 3 clock cycles to be executed

Decode one instruction and generates the required control signals

- This architecture was developed by Acorn, and employed up to ARM7
- The 3 stages are
 - Fetch
 - Decode
 - Execute
- Some instructions (e.g., those accessing the memory) require more than 3 clock cycles to be executed

3-stage ARM

- This architecture was the one originally developed by Acorn, and employed up to ARM7
- The 3 stages are
 - Fetch
 - Decode
 - Execute
- Some instructions (e.g., those accessing the memory) require more than 3 clock cycles to be executed

Execute one instruction:

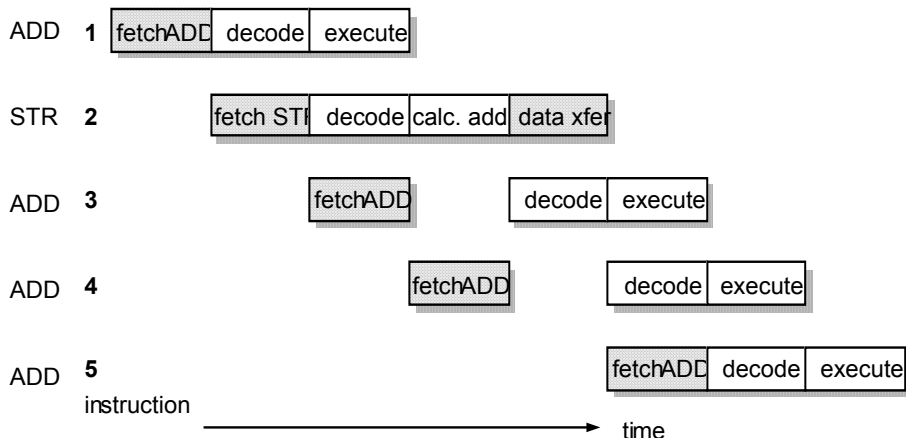
- The operands are read from registers
- One operand is possibly shifted
- The ALU generates the result
- The result is written in the destination register

- This architecture developed by
- The 3 stages
 - Fetch
 - Decode
 - Execute

For example, LOAD and STORE instructions require 2 clock cycle for execution (one for address computation, the other for memory access)

- Some instructions (e.g., those accessing the memory) require more than 3 clock cycles to be executed

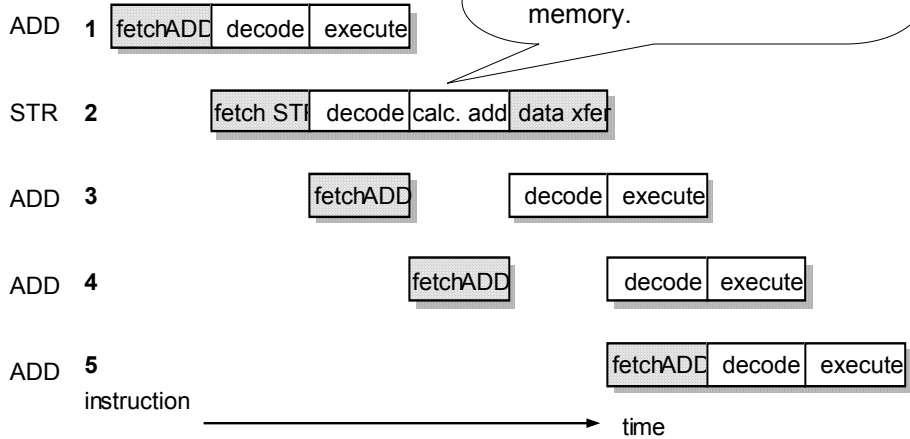
Pipeline behavior



Pipeline

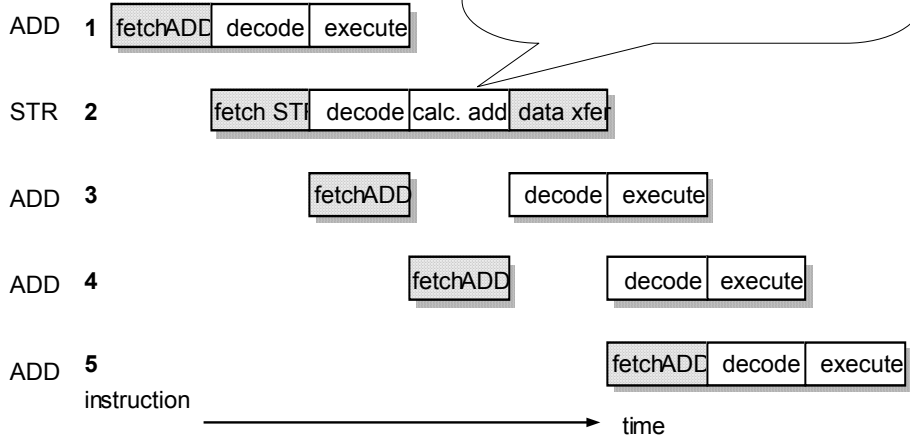
The STR instruction requires:

- 1 clock cycle to be fetched
- 1 clock cycle to be decoded
- 1 clock cycle to compute the memory address
- 1 clock cycle to access the memory.



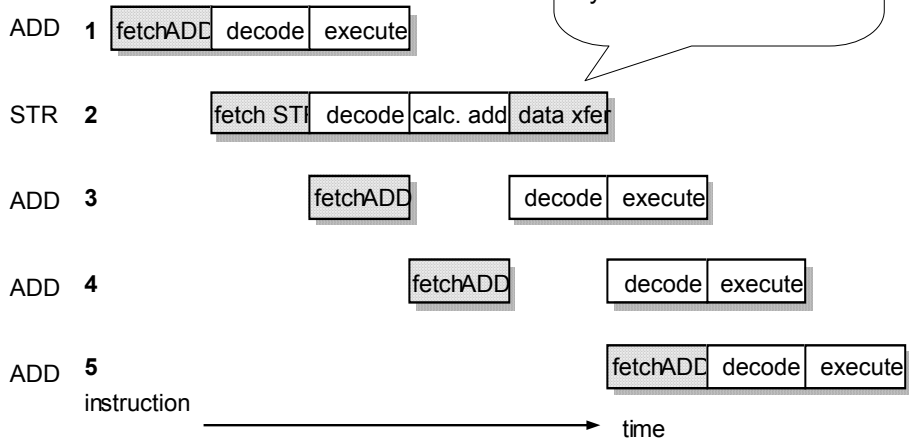
Pipeline

In this clock cycle the decode logic is still busy generating the control signal for accessing the memory in the following cycle.



Pipeline beh

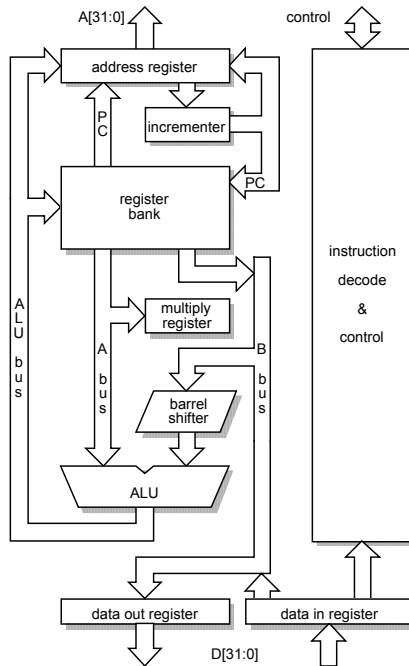
Only one memory access can be executed at any given clock cycle.



Branch instructions

- They always flush and refill the pipeline
- No delayed branch mechanism is supported

Architecture

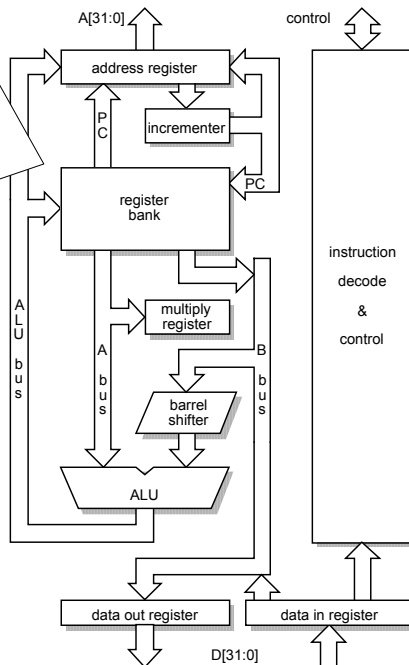


Matteo SONZA REORDA

Politecnico di Torino

13

It has two read ports and one write port. One additional read port and one additional write port are reserved for r15.



Matteo SONZA REORDA

Politecnico di Torino

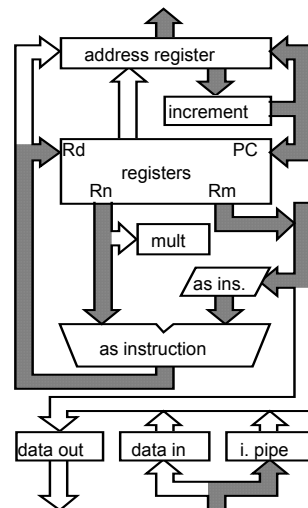
14

PC access

- If an instruction accesses the PC during the Execute stage, it reads a value which is incremented by 8 with respect to its proper address

Data processing reg-reg instruction execution

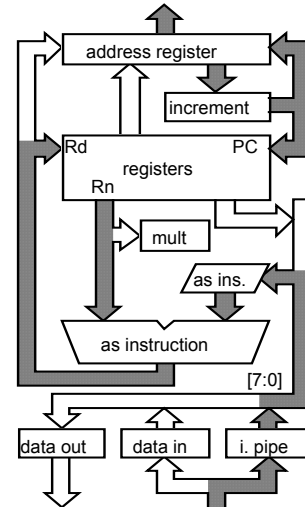
- Instruction *i* is executed:
 - Two operands are read from registers *Rn* and *Rm*
 - One operand is possibly rotated
 - The ALU generates the result
 - The result is written to register *Rd*
 - A further instruction is fetched from memory
 - The PC is updated



(a) register - register operations ;

Data processing reg-imm instruction execution

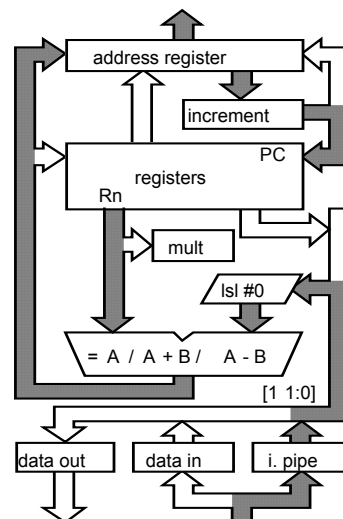
- Instruction i is executed:
 - One operand is read from register Rn, the other is an immediate
 - One operand is possibly rotated
 - The ALU generates the result
 - The result is written to register Rd
 - A further instruction is fetched from memory
 - The PC is updated



(b) r register - immediate operations

Data transfer instructions

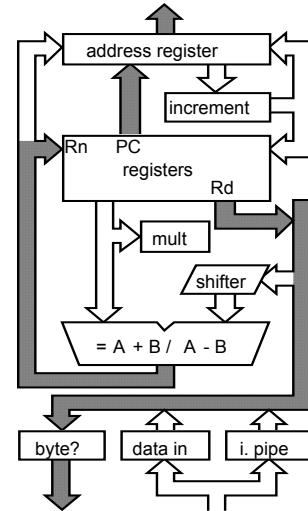
- They require two clock cycles for the Execute stage
- In the first, the address is computed using one register and one immediate



(a) 1st cycle - compute address

Data transfer instructions

- In the second clock cycle:
 - The memory is accessed
 - The destination register is updated (LDR instruction)
 - The index register is updated, if required by auto-indexing



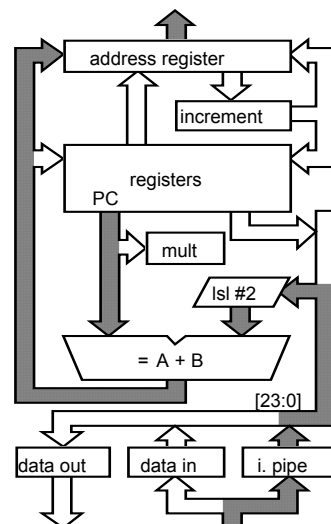
Matteo SONZA REORDA

Politecnico di Torino

(b) 2nd cycle - store data & auto-index

Branch instructions

- They first compute the target address, adding an immediate (shifted by 2 positions) to the PC
- Then, the pipeline is flushed and refilled



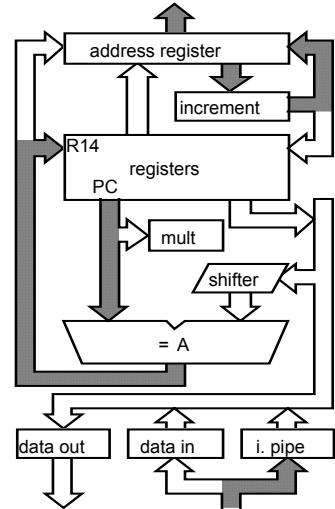
Matteo SONZA REORDA

Politecnico di Torino

(a) 1st cycle - compute branch target

Branch and link instructions

- In this case, a further clock cycle is required (while the pipeline is refilled) to save the return address in r14



Matteo SONZA REORDA

Politecnico di Torino

(b) 2nd cycle - save r return addr e21

Branch and link instructions

- They require a third clock cycle to adjust the saved address, so that it holds the address of the following instruction

Instruction speed

- Most instructions do require 3 clock cycles to execute
- Some instructions do require additional clock cycles
- The additional clock cycles may depend on specific values of the operands

Instruction speed

Instruction	Cycle count	Additional
Data Processing	1S	+ 1I for SHIFT(Rs) + 1S + 1N if R15 written
MSR, MRS	1S	
LDR	1S + 1N + 1I	+ 1S + 1N if R15 loaded
STR	2N	
LDM	nS + 1N + 1I	+ 1S + 1N if R15 loaded
STM	(n-1)S + 2N	
SWP	1S + 2N + 1I	
B,BL	2S + 1N	
SWI, trap	2S + 1N	
MUL,MLA	1S + mI	
CDP	1S + bI	
LDC,STC	(n-1)S + 2N + bI	
MCR	1N + bI + 1C	
MRC	1S + (b+1)I + 1C	

5-stage ARM

- The new architecture was adopted starting from ARM9
- It uses separate data and code memories (i.e., caches)
- The 5 stages are
 - Fetch
 - Decode
 - Execute
 - Buffer/data
 - Write-back
- The higher number of stages allows for a faster clock

Fetch one instruction from memory

- The new architecture was adopted starting from ARM9
- It uses separate data and code memories (i.e., caches)
- The 5 stages are
 - Fetch
 - Decode
 - Execute
 - Buffer/data
 - Write-back
- The higher number of stages allows for a faster clock

Decode the instruction and read the operands (up to 3) from registers

- The new ARM9
- It uses separate data and code memories (i.e., caches)
- The 5 stages are
 - Fetch
 - Decode
 - Execute
 - Buffer/data
 - Write-back
- The higher number of stages allows for a faster clock

- One operand is possibly shifted
- The ALU generates the result (or the address if a load/store instruction is considered)

- The new ARM9
- It uses separate data and code memories (i.e., caches)
- The 5 stages are
 - Fetch
 - Decode
 - Execute
 - Buffer/data
 - Write-back
- The higher number of stages allows for a faster clock

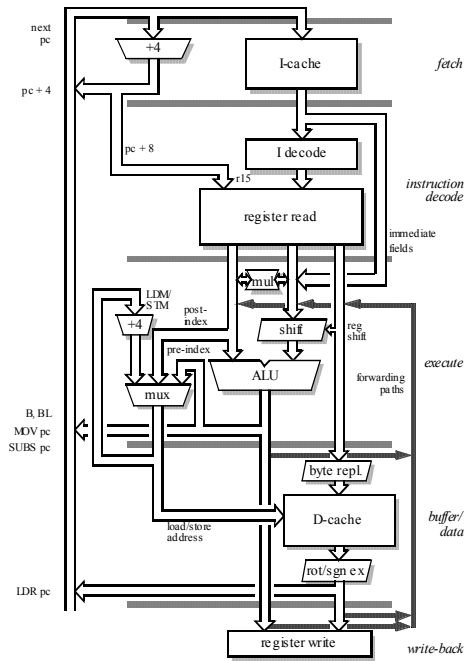
5-stage ARM

- The new ARM9
 - It uses separate caches)
 - The 5 stages are
 - Fetch
 - Decode
 - Execute
 - Buffer/data
 - Write-back
 - The higher number of stages allows for a faster clock
- Memory is accessed.
Only load/store instructions perform useful operations in this stage.

5-stage ARM

- The new ARM9
 - It uses separate caches)
 - The 5 stages are
 - Fetch
 - Decode
 - Execute
 - Buffer/data
 - Write-back
 - The higher number of stages allows for a faster clock
- Results (including values from memory) are written to the result register.

Architecture



Matteo SONZA REORDA

Politecnico di Torino

31

Data dependencies

- Their effect is limited by proper data forwarding logic
- When a data hazard can not be avoided, the processor is stalled

PC value

- Thanks to suitable connections, each instruction still reads a value from r15 which is incremented by 8 with respect to its address

Program compatibility

- 5-stage processors can execute the same binary code executed by 3-stage processors
- The same code can cause different stall situations in the 3-stage or 5-stage architectures

The ARM coprocessors

- The ARM instruction set can be extended by adding external coprocessors (up to 16)
- If coprocessors are absent, the corresponding instructions can be emulated in software through undefined instruction traps

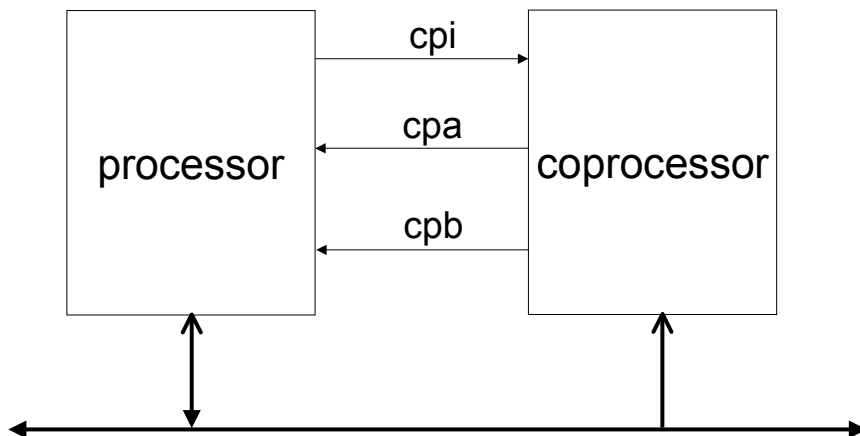
Coprocessor architecture

- Each coprocessor
 - May have up to 16 registers of any size
 - Has a load-store architecture, including instructions to move data
 - From one internal register to another
 - From one internal register to a ARM one, and viceversa
 - From one internal register to memory, and viceversa

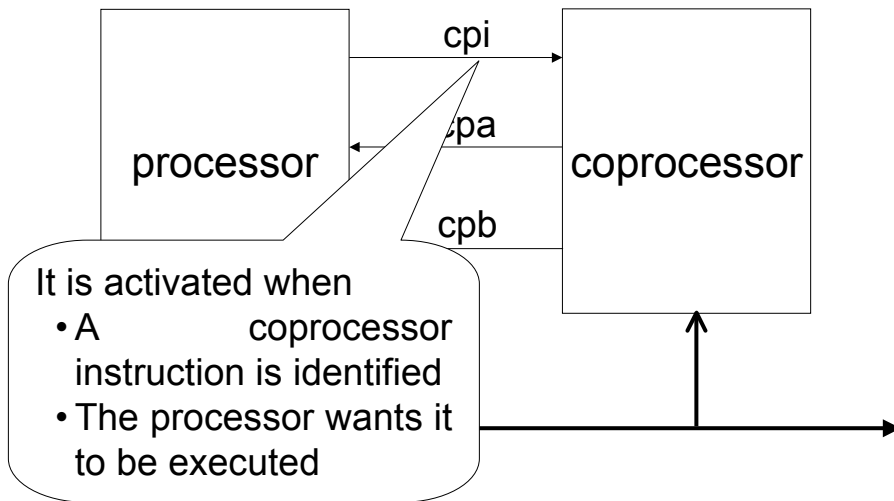
The ARM7TDMI coprocessor

- It is connected to the same data bus of the processor
- It continuously monitors the content of the bus, and fetches every instruction
- It contains an internal pipeline that mimics that in the processor

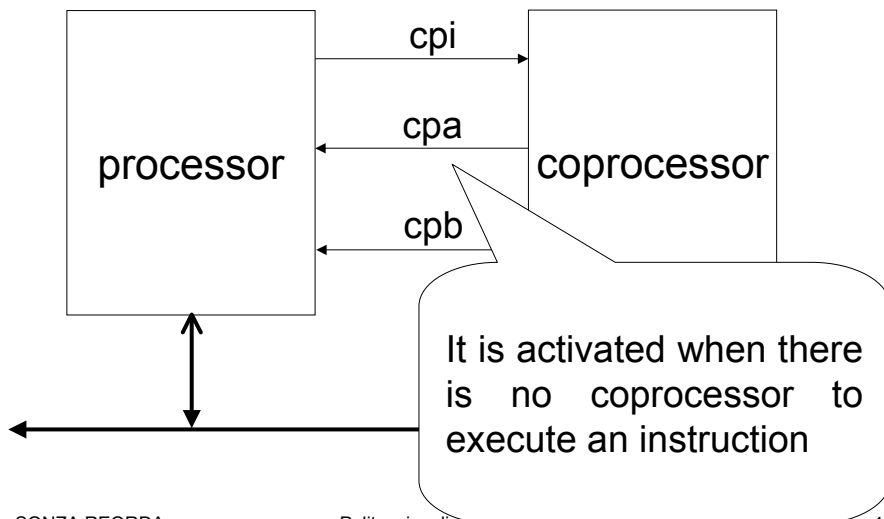
Processor-coprocessor interface

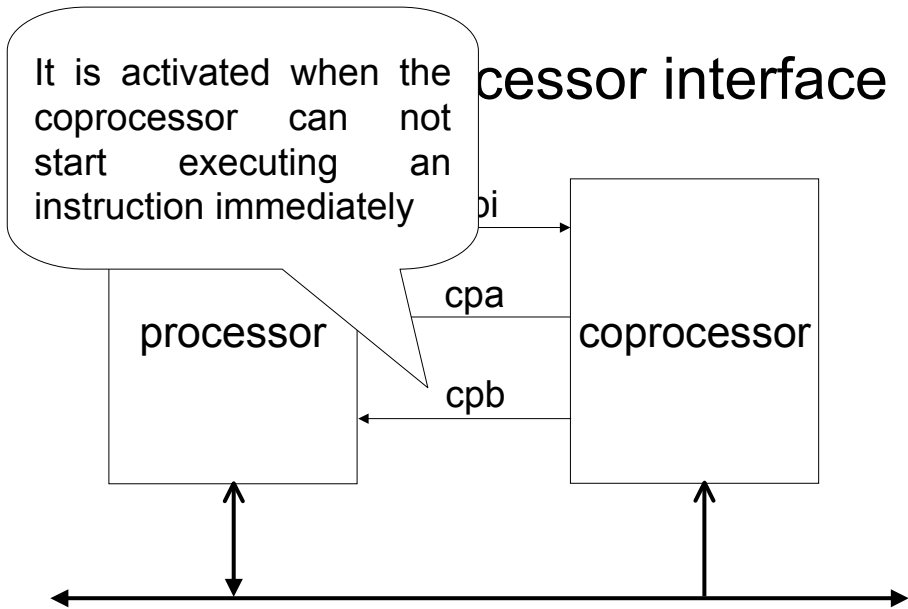


Processor-coprocessor interface



Processor-coprocessor interface





Processor-coprocessor handshaking

- Once an instruction has entered the processor and coprocessor pipelines, one out of the four following possibilities may arise:
 - The ARM may decide not to execute it (e.g., because it is conditioned, and the condition was false). *cpi* is not activated. Both the processor and the coprocessor will discard the instruction.

Processor-coprocessor handshaking

- Once an instruction has entered the processor and coprocessor pipelines, one out of the four following possibilities may arise:
 - The ARM may decide to execute it (*cpi* is activated), but the corresponding coprocessor does not exist (*cpa* remains active). The undefined instruction trap is triggered, and the instruction is possibly emulated.

Processor-coprocessor handshaking

- Once an instruction has entered the processor and coprocessor pipelines, one out of the four following possibilities may arise:
 - The ARM decides to execute it (*cpi* is activated), the corresponding coprocessor exists (*cpa* is deactivated), but it is not ready to start execution (*cpb* is active). The processor starts waiting, stalling the instruction stream. Only interrupt requests are served.

Processor-coprocessor handshaking

- Once an instruction has entered the processor and coprocessor pipelines, one out of the four following possibilities may arise:
 - The ARM decides to execute it (*cpi* is activated), the corresponding coprocessor exists (*cpa* is deactivated), and it is ready to immediately start execution (*cpb* is deactivated). The instruction starts its execution.

Data transfers

- Since coprocessors are not connected to the address bus, when memory must be accessed by a coprocessor instruction, the processor generates addresses.