Xi'an Jiaotong-Liverpool University

西交利物浦大學

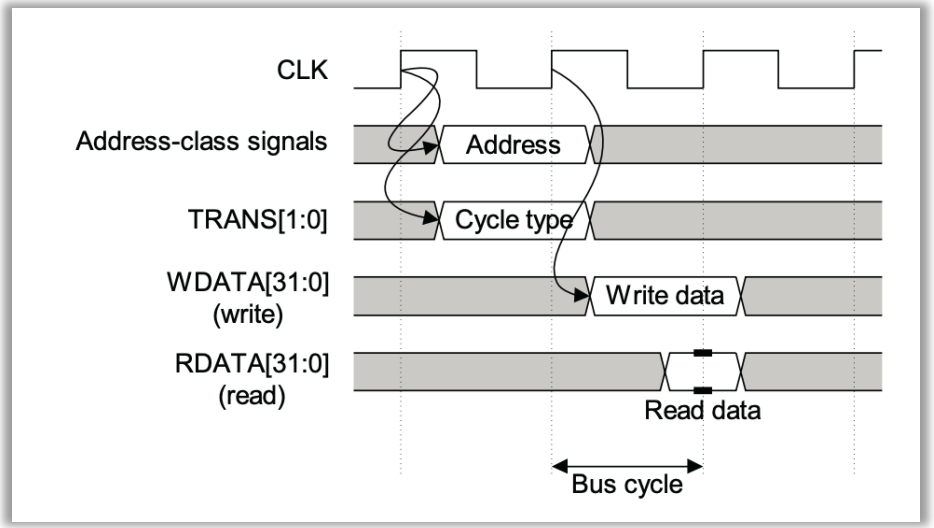
ARM7TDMI Pipeline

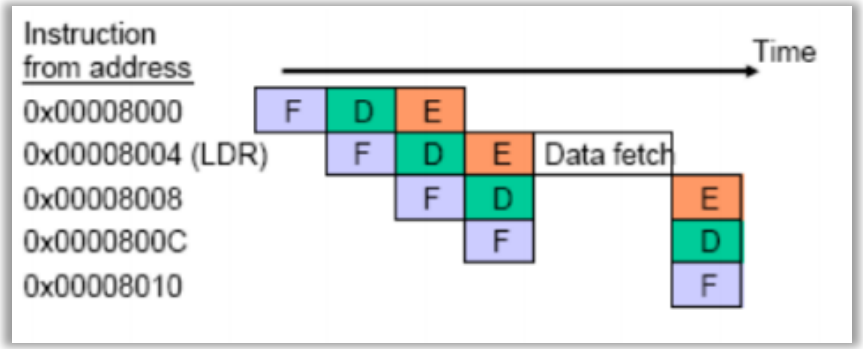
Jianjun Chen

Pipeline

Bus Cycle Types

Instruction Pipelines



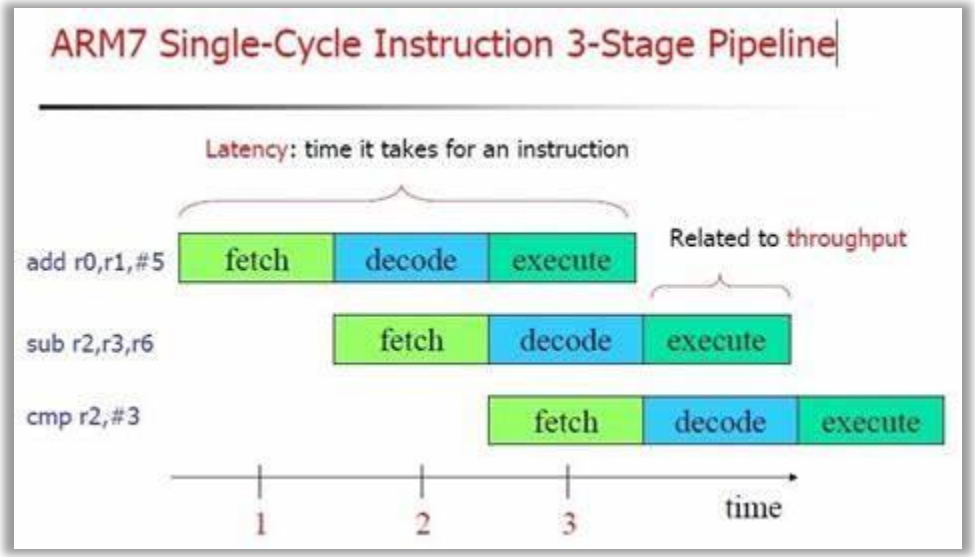


Instruction Pipelines

- Instruction pipeline is an important feature of all modern microprocessors.
- The basic idea is to split the processor instructions into a series of small independent stages. Each stage is designed to perform a certain part of the instruction.
- The ARMv7 microprocessor has a three stage pipeline:
  - Fetch
  - Decode
  - Execute

Instruction Pipelines

- In a three stage pipeline, the CPU can simultaneously execute an instruction, decode the next instruction and fetch the next instruction.
  - Each cycle can do one stage.
  - Cycle: The time needed for one simple processor operation.



Execution Cycle Times

- Simple data processing instructions only take 1 cycle to **execute**:

```
ADD R0, R1, R2
```

- More complicated ones may take 2 cycles:

```
ADD R0, R1, R2, ROR R3
```

- `LDR` normally takes 3 cycles to execute.
- `STR` takes 2 cycles to execute.

- When the condition is NOT met, all instructions take one cycle.

```
ADDEQ R0, R1, R2
```

Bus Cycle Types

- Pipeline involves accessing the main memory or external devices.
- The **bus cycle** is the time required to make a single read or write transaction.
- The ARM7TDMI-S processor supports four basic types of bus cycle:
  - **`N`**: Nonsequential cycle
  - **`S`**: Sequential cycle
  - **`I`**: Internal cycle
  - **`C`**: Coprocessor register transfer cycle

Bus Cycle Types

- Nonsequential cycle:
  - The CPU requests a transfer to, or from an address which is unrelated to the address used in the preceding cycle.
- Sequential cycle
  - The CPU requests a transfer to or from an address that is either one word or one halfword greater than the address used in the preceding cycle.
- Internal cycle
  - The CPU does not require a transfer because it is performing an internal function and no useful prefetching can be performed at the same time.
- Coprocessor register transfer cycle
  - The CPU uses the data bus to communicate with a coprocessor but does not require any action by the memory system

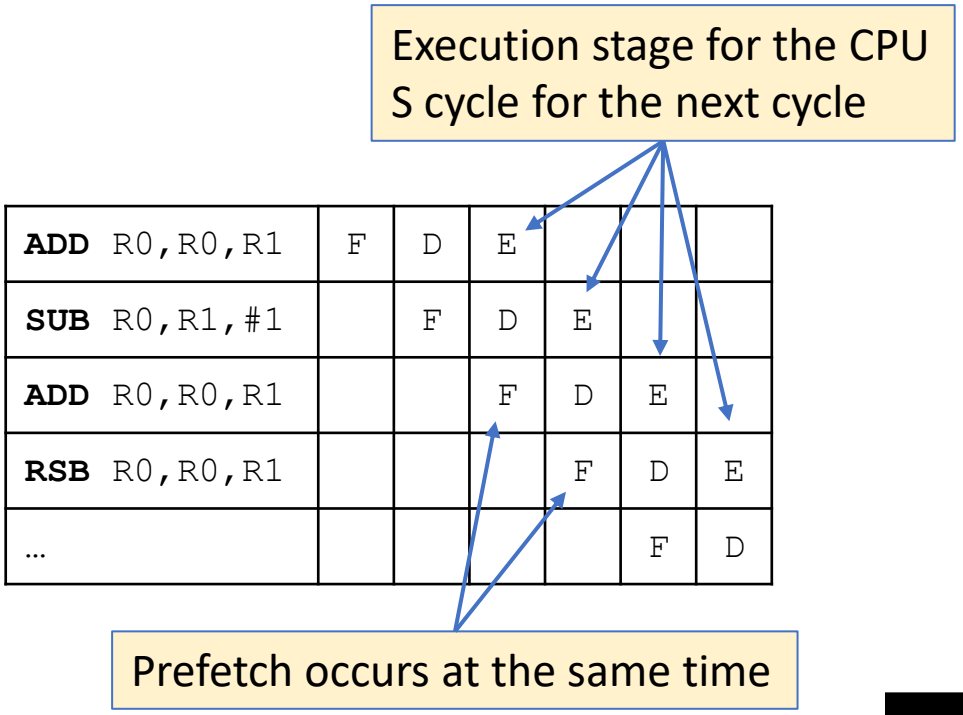
Instruction Cycle Count

Processing Type	Cycles
Any unexecuted (condition code fails)	+S
Normal Data Processing	+S
Data Processing with register specified shift	+I +S
Data Processing with PC written	+N +2S
Data Processing with register specified shift and PC written	+I +N +2S
LDR	+N +I +S
LDR into PC	+N +I +N +2S
STR	+N +N
LDM (n is the number of words transfered)	+N +(n–1)S +I +S
LDM with PC	+N +(n–1)S +I +N +2S
STM	+N +(n–1)S +I +N
B, BL	+N +2S

+S means the **next cycle** is sequential cycle

Data Processing Instructions

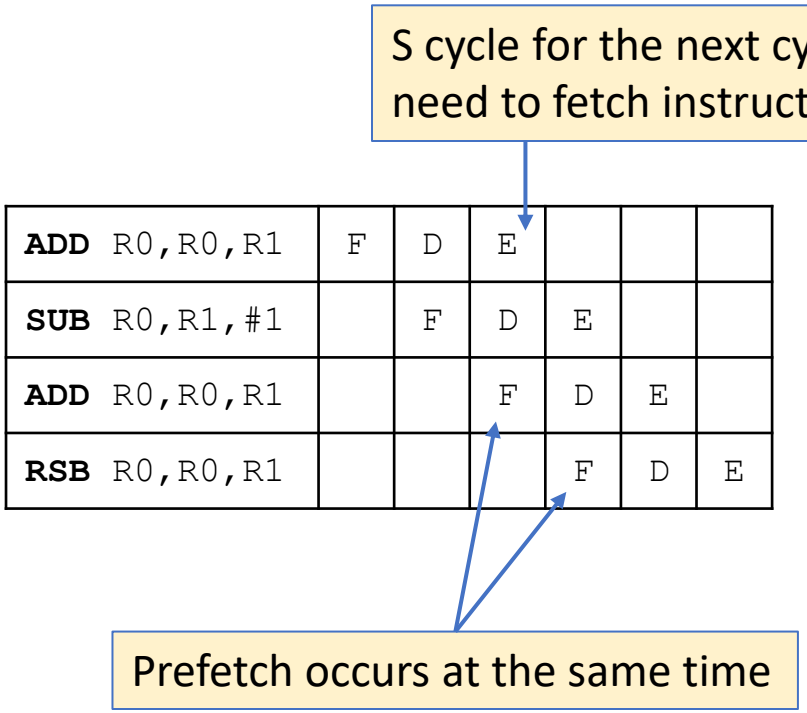
- An instruction prefetch occurs at the same time as the data operation, and the PC is incremented.



Processing Type	Cycles
Normal Data Processing	+S

“S cycle for the next cycle”

- The data operation only takes one cycle to execute.
- In the perspective of the system bus, the next step is to fetch a new instruction.



Processing Type	Cycles
Normal Data Processing	+S

Pipelines - Optimum Operation

- A pipeline operates optimally if the instructions to be executed are:
  - In consecutive memory locations and,
  - No conflict occurs on the data bus.
- When this is the case the microprocessors operates at one clock cycle per instruction (1 CPI).
- The best performance cannot be achieved if a branch or load instruction is executed or if an interrupt is serviced.
  - Branch and interrupt leads to non-consecutive memory location
  - Load uses the data bus, which prevents the pipeline from fetching future instructions.

Data Processing Instructions

- When a register specifies the shift length, an additional data path cycle occurs before the data operation.
  - The instruction prefetch occurs during this first cycle

Next cycle is I cycle because the instruction is not finished. Need to do LSL then ADD.

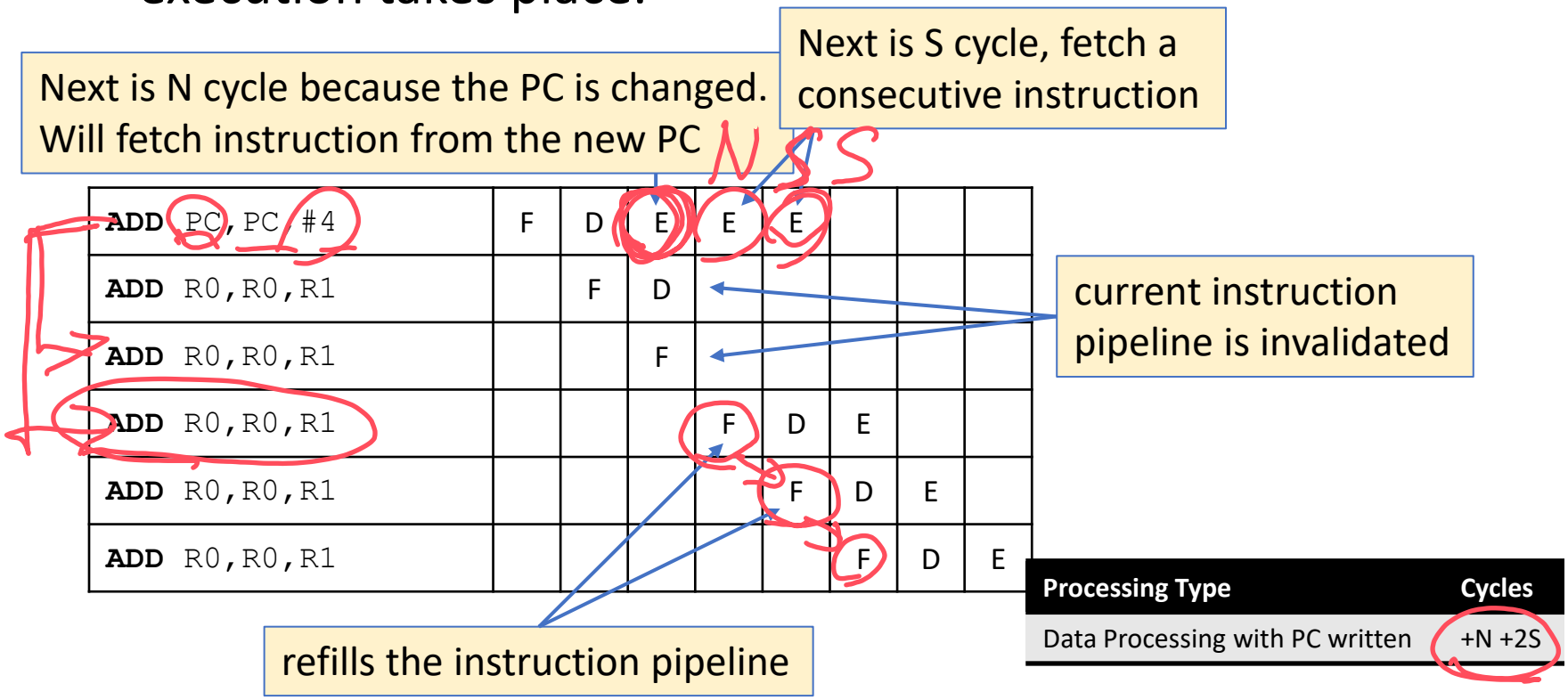
Next cycle is S cycle because a new instruction should be fetched

ADD R0,R0,R1	F	D	E						
SUB R0,R1,#1		F	D	E					
ADD R0,R0,R1,**LSL** R3			F	D	E	E			
ADD R0,R0,R1				F	D		E		
ADD R0,R0,R1					F		D	E	
ADD R0,R0,R1							F	D	E

Processing Type	Cycles
Data Processing with register specified shift	+I +S

Data Processing Instructions

- When the PC is the destination, the CPU writes the result to the PC, the current instruction pipeline is invalidated.
- The CPU then refills the instruction pipeline before any more execution takes place.



ADD PC,PC,#4	F	D	E	E	E			
ADD R0,R0,R1		F	D					
ADD R0,R0,R1			F					
ADD R0,R0,R1				F	D	E		
ADD R0,R0,R1					F	D	E	
ADD R0,R0,R1						F	D	E

Processing Type	Cycles
Data Processing with PC written	+N +2S

Data Processing Instructions

"N cycle: The CPU requests a transfer to, or from an address which is unrelated to the address used in the **preceding** cycle."

"S cycle: The CPU requests a transfer to or from an address that is either one word or one halfword greater than the address used in the **preceding** cycle."

Next is N cycle because the PC is changed. Will fetch instruction from the new PC

Next is S cycle, fetch a consecutive instruction

ADD PC,PC,#4	F	D	E	E	E			
ADD R0,R0,R1		F	D					
ADD R0,R0,R1			F					
ADD R0,R0,R1				F	D	E		

current instruction pipeline is invalidated

Branch, Branch with Link

- B and BL takes 3 cycles:

- The branch instruction calculates the branch destination while performing a prefetch from the current PC.
  - This prefetch is always done by the pipeline. Cannot be avoided

- The CPU performs a Fetch from the branch destination. The return address is stored in r14 if the link bit is set.

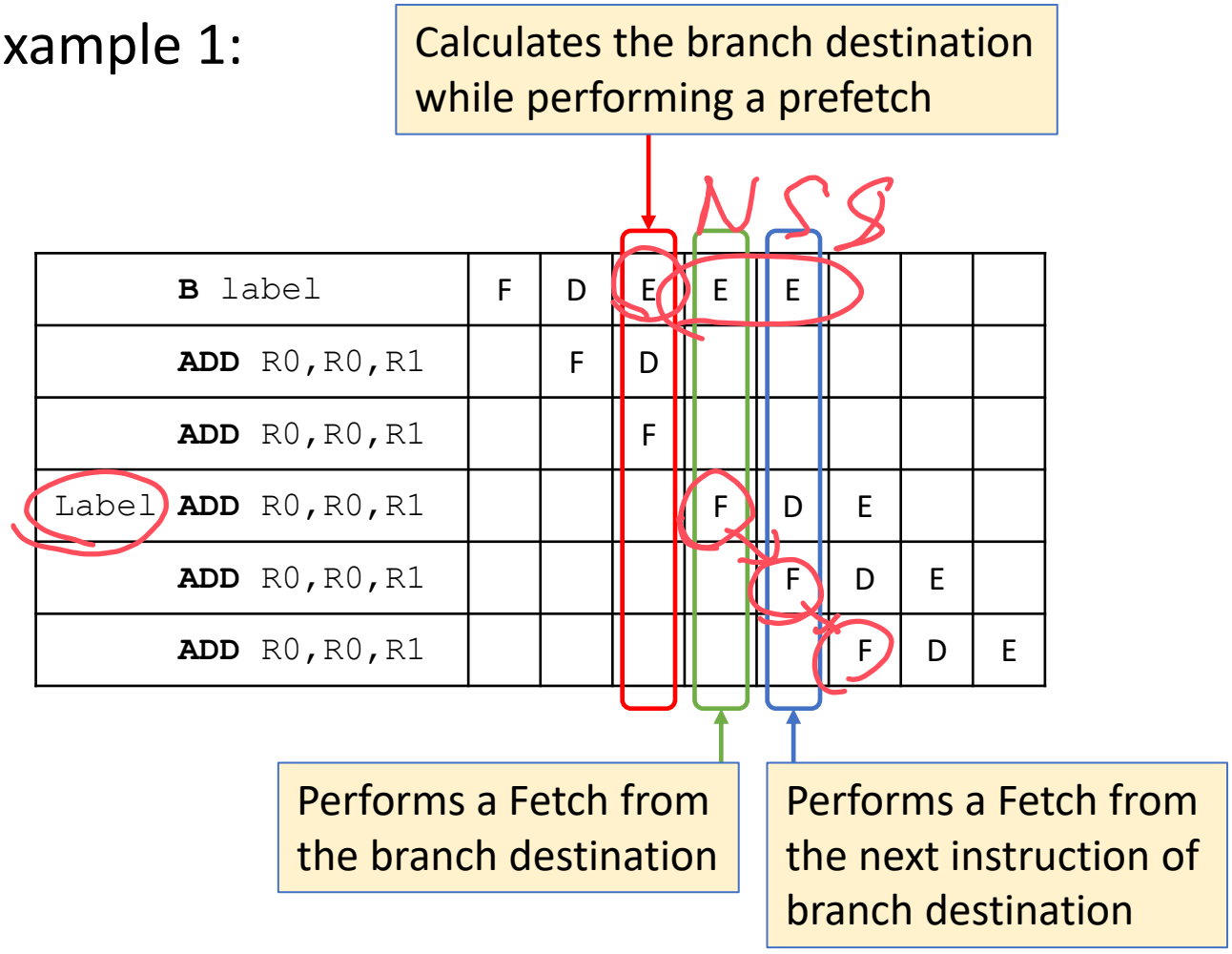
- The CPU performs a Fetch from the destination + i, refilling the instruction pipeline. When the instruction is a branch with link, r14 is modified.

Processing Type	Cycles
B, BL	+N +2S

Branch, Branch with Link

Processing Type	Cycles
B, BL	N +2S

- Example 1:



Branch, Branch with Link

Processing Type	Cycles
B, BL	N +2S

- Example 2:

Calculates the branch destination while performing a prefetch

B label	F	D	E	E	E					
Label **ADD** R0,R0,R1		F	D	F	D	E				
ADD R0,R0,R1			F		F	D	E			
ADD R0,R0,R1						F	D	E		
ADD R0,R0,R1							F	D	E	
ADD R0,R0,R1								F	D	E

Performs a Fetch from the branch destination

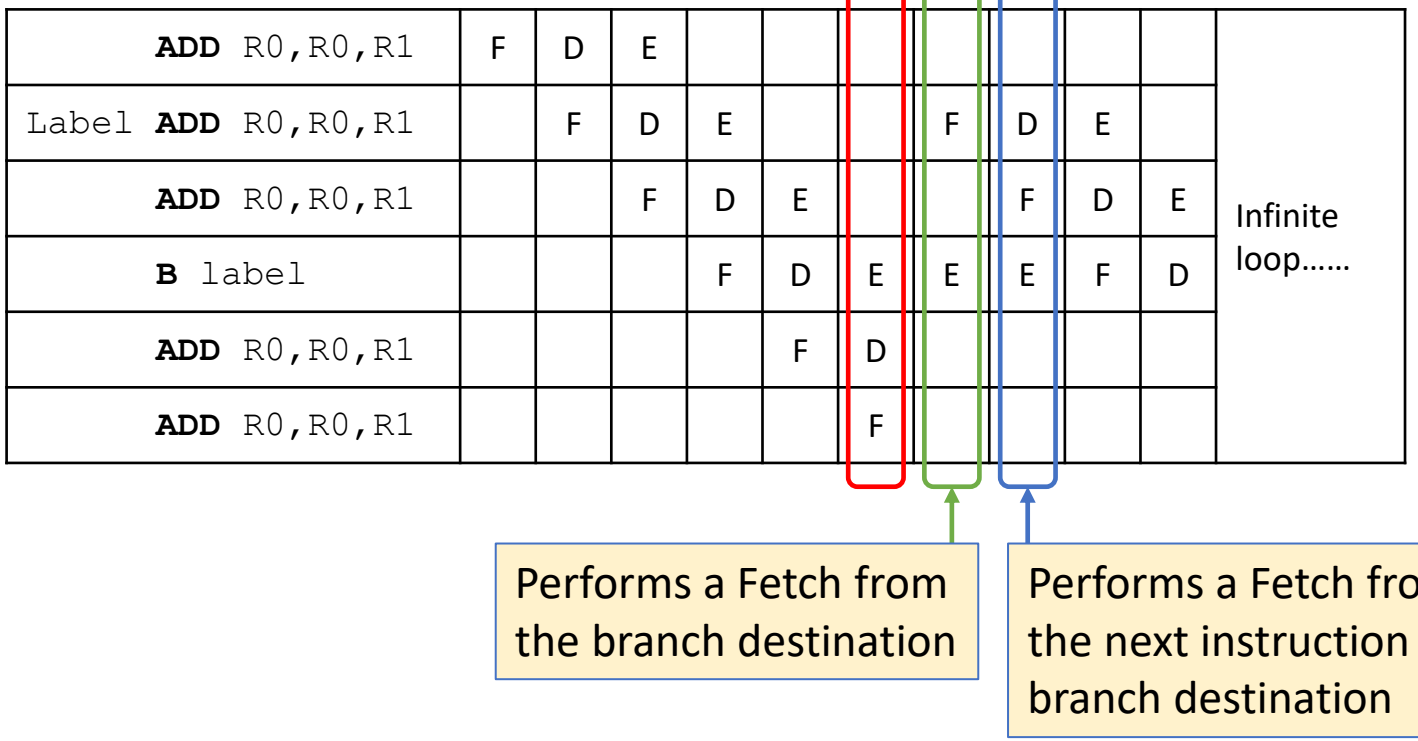
Performs a Fetch from the next instruction of branch destination

Branch, Branch with Link

Processing Type	Cycles
B, BL	N +2S

- Example 3:

Calculates the branch destination while performing a prefetch



Performs a Fetch from the branch destination

Performs a Fetch from the next instruction of branch destination

Question

- How many clock cycles do the following instructions take to execute if the value in r0 is
  - not equal to 0 and
  - equal to 0?

```
        MOVS r0, r0
        BNE cont      ; branch if not zero
        ADD r3, r4, r5
        EOR r2, r1, r4
cont    SUB r6, r7, #42
```

*NE tests whether the zero bit is set

Answer

- Not equal to 0
  - The branch is taken so that needs 3 clock cycles.
  - The MOV and SUB instructions take 1 each
  - so 5 + 2 clock cycles are required in total.
- Equal to 0
  - The branch is not taken so that only needs one clock cycle.
  - The other instructions each take 1 so again 5 + 2 clock cycles are needed.

```
        MOVS r0, r0
        BNE cont      ; branch if not zero
        ADD r3, r4, r5
        EOR r2, r1, r4
cont    SUB r6, r7, #42
```

Question

- How many clock cycles do the following instructions take to execute if the value in r0 is
  - not equal to 0 and
  - equal to 0?

```
MOVS r0, r0
ADDEQ r3, r4, r5
EOREQ r2, r1, r4
SUB r6, r7, #42
```

Answer

- The ADDEQ and EOREQ instructions need one clock cycle no matter if they are executed or not.
- So each instruction takes one clock cycle and 4+2 clock cycles are needed in total.
- Faster than the previous set of instructions.

Load Register (LDR)

Processing Type	Cycles
LDR	+N +I +S
LDR into PC	+N +I +N +2S

- LDR normally takes 3 cycles to execute:
  - The CPU calculates the address to be loaded.
  - The CPU fetches the data from memory and performs the base register modification (if required).
  - The CPU transfers the data to the destination register and do prefetch.

- If PC is the destination, 2 more steps are needed
  - The CPU calculates the address to be loaded.
  - The CPU fetches the data from memory.
  - The CPU transfers the data to the destination register (no prefetch).
  - Performs a Fetch from the new PC value.
  - Performs a Fetch from the next instruction of the new PC. (PC + 4)
    - If in Thumb mode, it will be PC+2 as thumb instructions are 16 bits long.

Load Register (LDR)

Processing Type	Cycles
LDR	+N +I +S
LDR into PC	+N +I +N +2S

- Example 1

Calculates the address to be loaded.
Next cycle is N cycle: load data from memory. (nonsequential access)

ADR R1,nums	F	D	E						
LDR R0,[R1]		F	D	E	E	E			
ADD R0,R0,R1			F	D			E		
ADD R0,R0,R1				F			D	E	
ADD R0,R0,R1							F	D	E
ADD R0,R0,R1								F	D	E

Q: Can this "Decode" happen at any time between F and E?

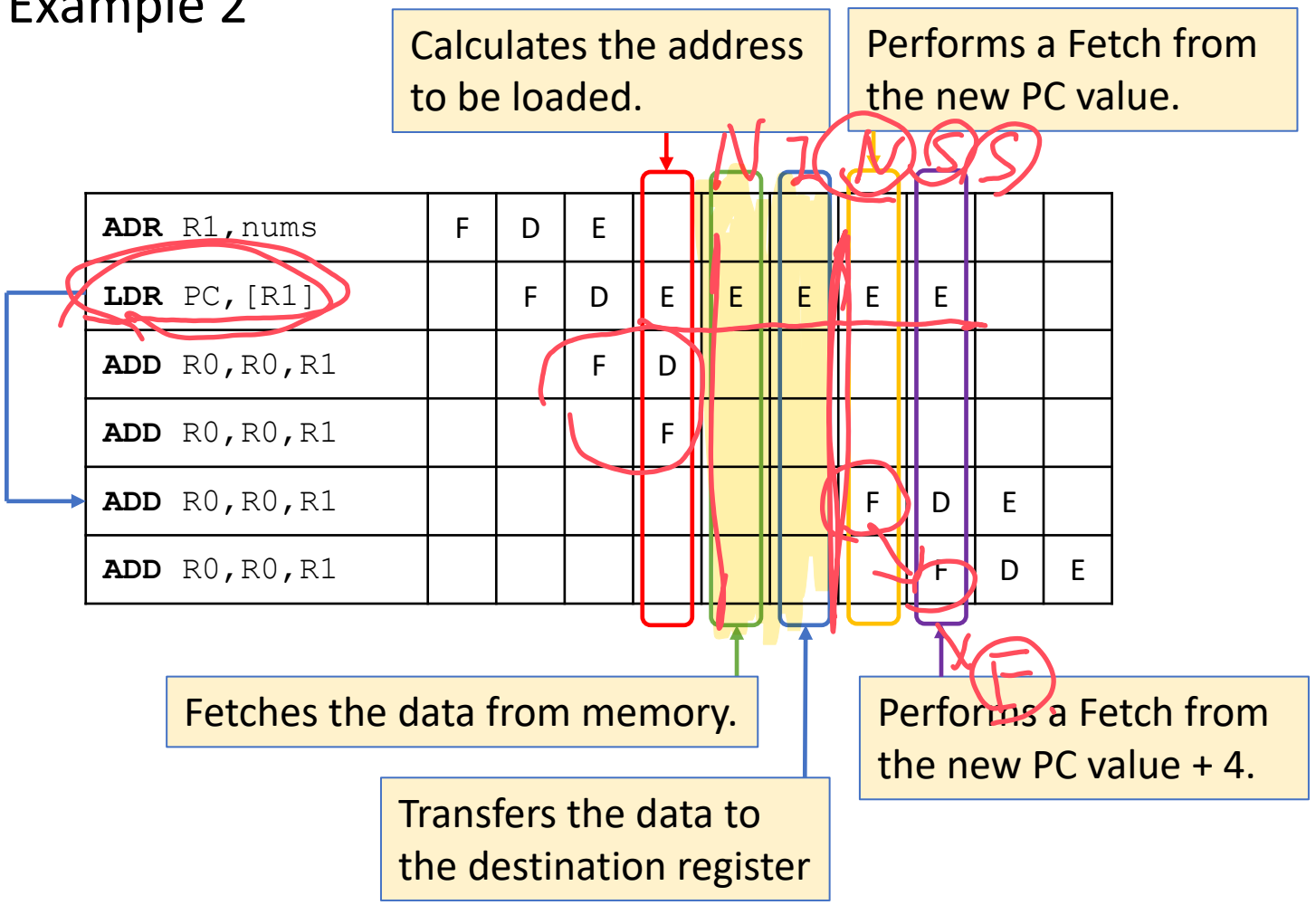
Fetches the data from memory.
Next cycle is I cycle.

Transfers the data to the destination register and do prefetch.
Next cycle is S cycle: fetch next instruction

Load Register (LDR)

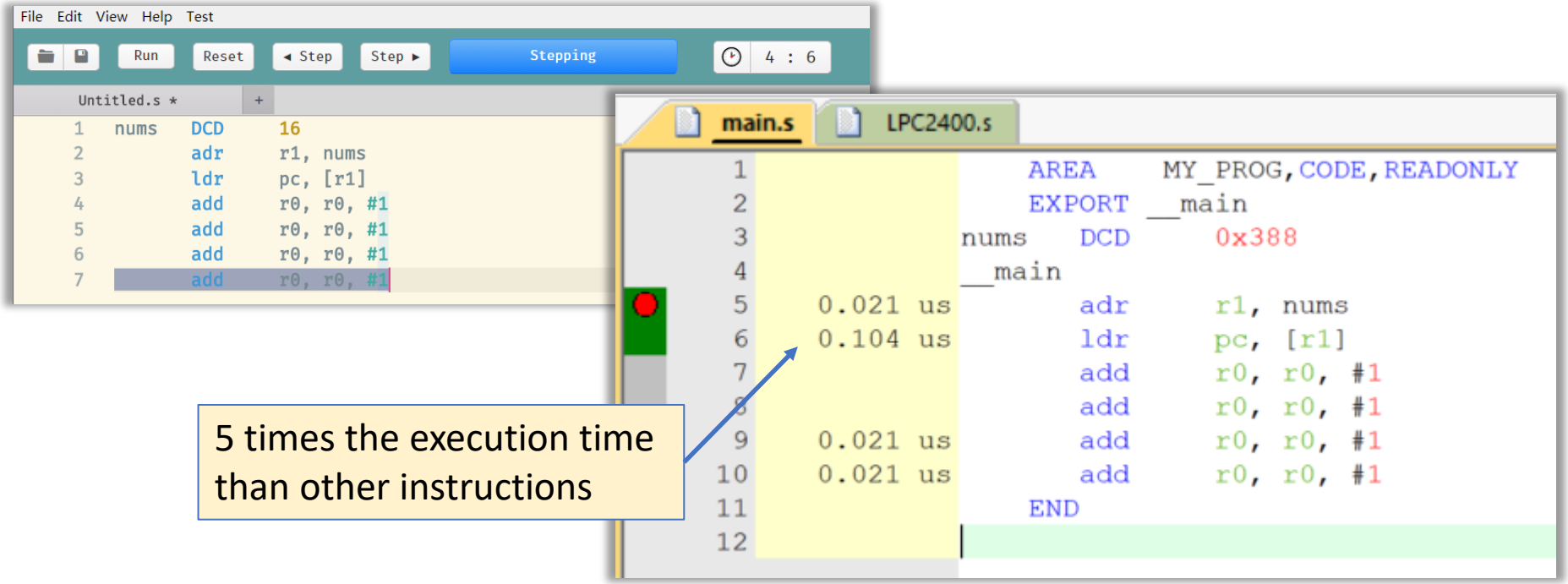
Processing Type	Cycles
LDR	+N +I +S
LDR into PC	+N +I +N +2S

- Example 2



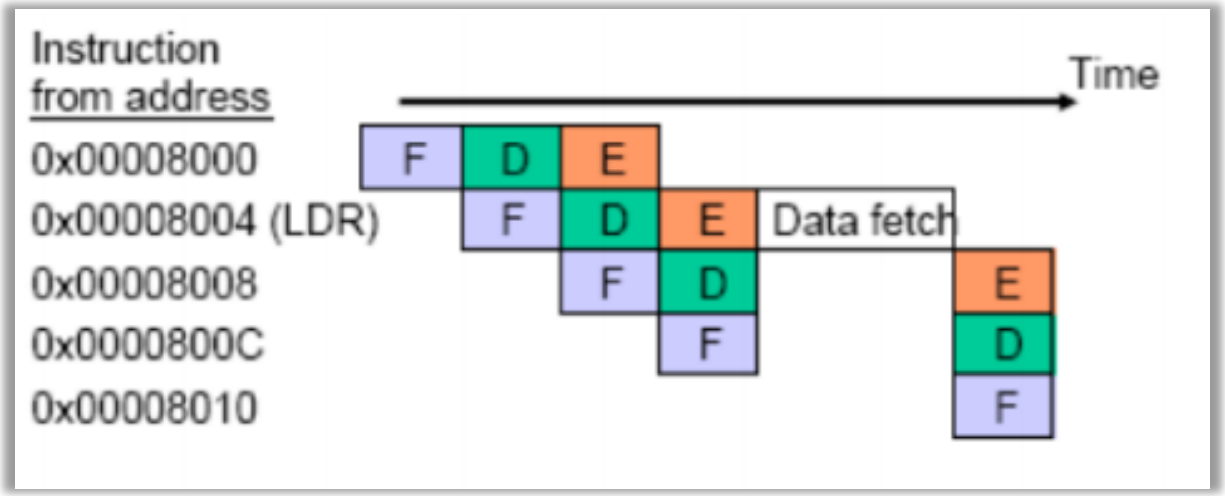
Load Register (LDR)

- The number of cycles in the previous example is not calculated correctly in VisUAL.
- Verify using Keil: use Debug function and turn on “show time”
  - Menu -> Debug -> Execution Profiling -> show time



Load and Store Instructions

- Load and store instructions use the data bus to pass data to or from memory so that the data bus cannot be used to fetch an instruction at the same time.



Eliminating Bus Conflicts

- The data fetch uses two clock cycles so that in the previous example only three instructions are executed in five clock cycles - 1.66 CPI.

- Bus conflicts could be eliminated by using two separate data buses; one for instructions and one for load and store data - this is called a **Harvard architecture**.

- A microprocessor, such as the ARM7, that uses one data bus for both instructions and data is said to have the **Von Neumann architecture**.

Von Neumann or Harvard?

Von Neumann

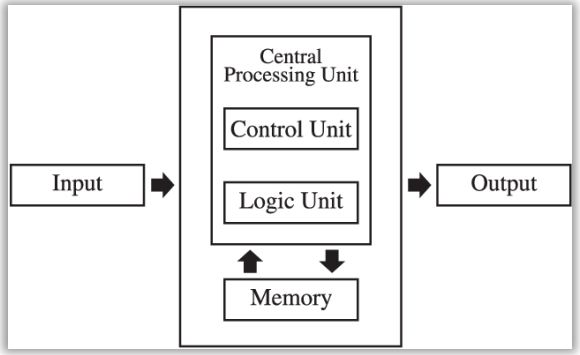
- ARM7 (Von Neumann) has an average CPI of 1.9

Harvard

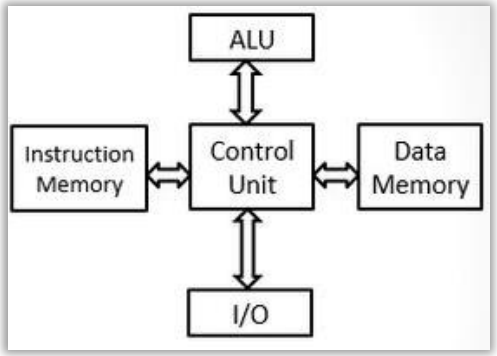
- ARM9 (Harvard) has an average CPI of 1.5

depends on the program being executed

- Bus conflicts when reading/writing memory.
- Less complicated design than Harvard.



- Fewer lost clock cycles due to bus conflicts
- Harvard has greater complexity.



3 Stage VS 5 Stage Pipeline?

- The maximum clock frequency of the ARM7 is limited by a bottleneck during the execute stage of the 3 stage pipeline.
- The ARM9 has a 5 stage pipeline so that there is less work in each stage of the pipeline and therefore a higher maximum clock frequency can be achieved for the same technology.
  - MIPS also use a 5 stage pipeline.
- However more stages require more power because more circuits are functioning simultaneously

ARMv7 VS ARMv9

Processor:	ARM720T	ARM922T
Architecture:	Von Neumann	Harvard
Pipeline:	3 stage	5 stage
Transistors in core:	74,209	111,000
Area including cache:	2.4 mm^2	3.2 mm^2
Clock:	0 to 100 MHz	0 to 250 MHz
Power:	upto 20 mW	upto 62.5 mW
mW/MHz	0.2	0.25

Homework

- Write a small program in VisUAL and try to work out the number of clock cycles it takes too run by hand.
- Step to the last line of code and do NOT finish the execution.