TMS320C3X Floating Point DSP

Microcontrollers & Microprocessors Undergraduate Course
Isfahan University of Technology – Oct 2010
By : Mohammad Sadegh Sadri

DSP

- DSP : Digital Signal Processor
- Why A DSP?
  - Example
    - Voice Recorder
- DSP
  - Low cost
  - High Performance
  - Dedicated to processing applications
  - Simple Architecture
    - Real Time Systems

DSP Manufacturers

- Texas Instruments
- Analog Devices
- Motorola
  - Now : Freescale
- …

Digital Signal Processing in GP CPUs

- By the time passed
  - More DSP functionality added to General Purpose CPUs
- Today CPUs
  - Contain huge hardware
  - For signal processing calculations
- Example
  - Streaming SIMD Extension

Two Major DSP Categories

- Floating Point
  - Higher Silicon Area is required for implementation of Floating Point Unit
  - Higher Prices
  - Lower Clock Frequencies
  - Lower Level of parallelism
  - Easy Algorithm Design

- Fixed Point DSP
  - Very difficult algorithm design
    - Designer should take care of losing data
  - Smaller silicon Area
  - Higher clock frequency
  - Smaller price
  - Higher Level of parallelism
  - Totally : higher level of performance

Texas Instruments

- TMS320C10/C25
- TMS320C30/31/32/40
  - Floating Point
- TMS320C5x
  - Fixed Point Ultra Low Power
- TMS320C6x : MIMD architecture
  - TMS320C67x : Floating Point
  - TMS320C62x/64x : Fixed Point
    - Very high levels of performance
- …

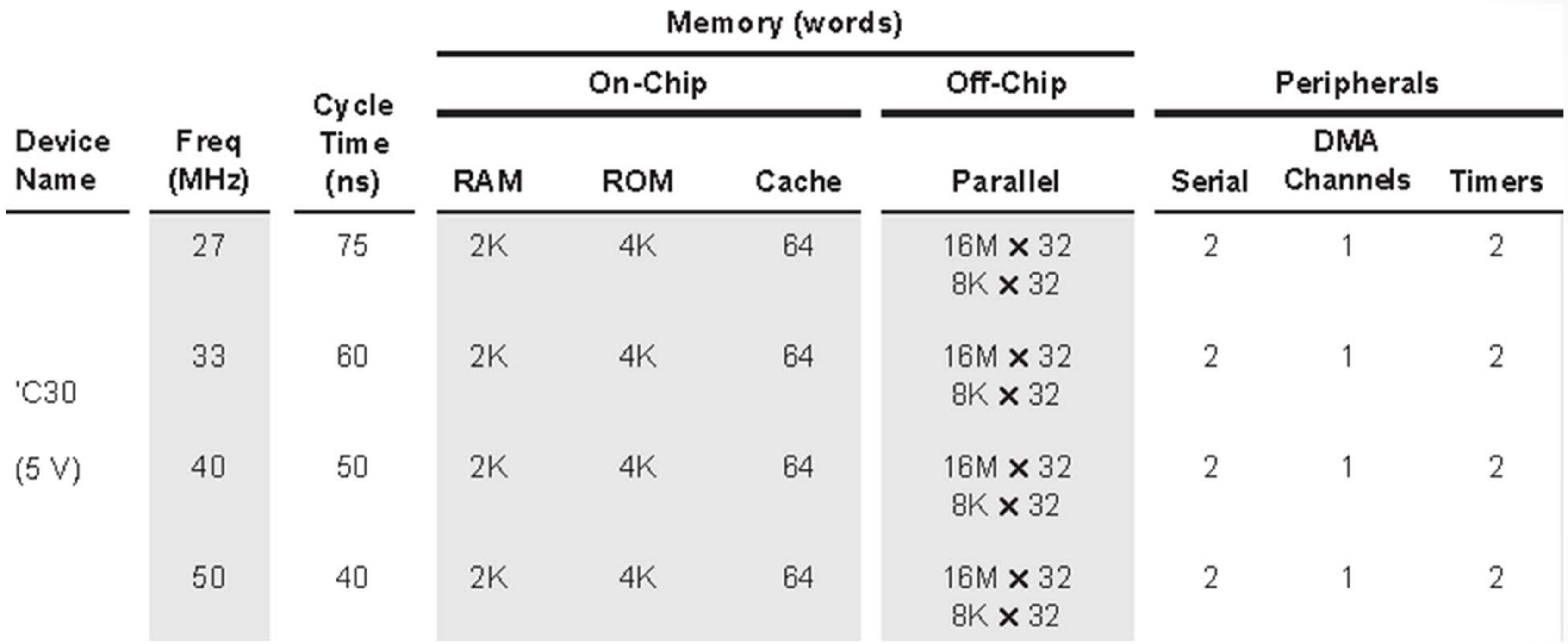
TMS320C3x DSPs

- 1988
- Harvard Architecture
- Floating point computations
- Addressing Range: 24 Bits (16Mbytes)
- 3 Families
  - TMS320C30
  - TMS320C31
  - TMS320C32
- C30 : contained a boot ROM
- C31/C32 : contained Boot Loader instead

Boot Loader

- What is a boot loader?
  - An application
  - Which is the first application executed by CPU
  - Responsible for Loading the main application into system memory and starting it
- Boot Loader is usually small
- Boot Loader contains the first instructions that are executed by CPU
  - Boot loader design is tricky
- Boot loader is usually stored on a kind of flash memory

TMS320C30 Family



			Memory (words)				Peripherals		
			On-Chip			Off-Chip			
Device Name	Freq (MHz)	Cycle Time (ns)	RAM	ROM	Cache	Parallel	Serial	DMA Channels	Timers
'C30 (5 V)	27	75	2K	4K	64	16M × 32 8K × 32	2	1	2
	33	60	2K	4K	64	16M × 32 8K × 32	2	1	2
	40	50	2K	4K	64	16M × 32 8K × 32	2	1	2
	50	40	2K	4K	64	16M × 32 8K × 32	2	1	2

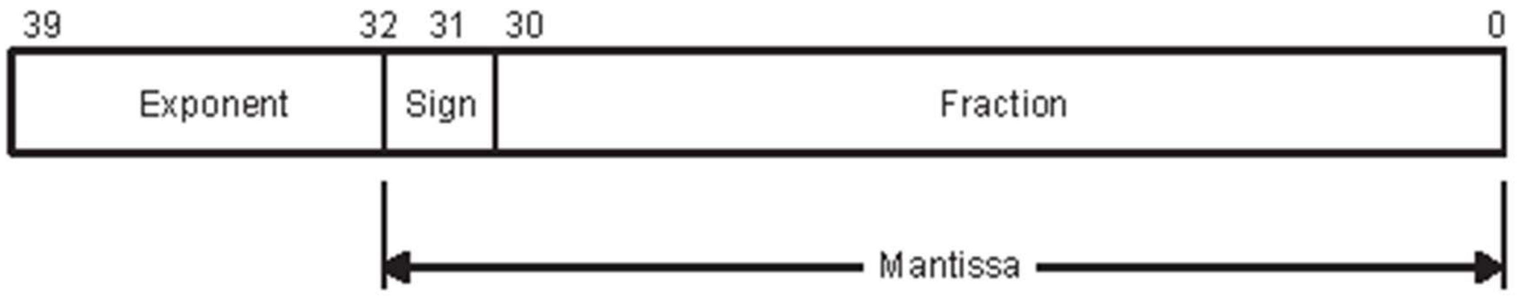
TMS320C32 Internal Block Diagram

- SPRU031e
  - Page 43
- Description of Buses
  - Program buses
    - PADDR, PDATA
  - Data buses
    - 2 Data memory accesses every machine cycle
    - DDATA, DADDR1, DADDR2, CPU1, CPU2
    - REG1, REG2 : no connected to memory. (Internal Bus)
  - DMA buses
    - DMAADDR, DMADATA

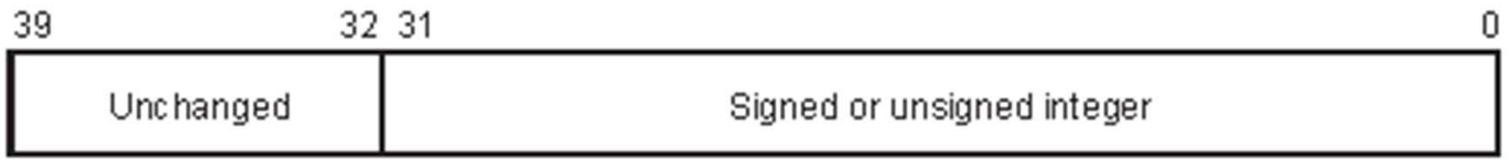
Registers

- R0 ~ R7
  - 8 Extended Precision Registers (40Bits)
- Exponent and Mantissa : 2's complement

Extended-Precision Register Floating-Point Format



Extended-Precision Register Integer Format



Extended Precision Range

Most positive:	$x = (2 - 2^{-23}) \times 2^{127} = 3.4028234 \times 10^{38}$
Least positive:	$x = 1 \times 2^{-127} = 5.8774717 \times 10^{-39}$
Least negative:	$x = (-1 - 2^{-23}) \times 2^{-127} = -5.8774724 \times 10^{-39}$
Most negative:	$x = -2 \times 2^{127} = -3.4028236 \times 10^{38}$

Auxiliary Registers

- 8 Registers AR0~AR7
  - 32Bits
  - Mainly used for Address Generation
  - Can be used as 32Bits General Purpose Registers
- Index Registers IR0 and IR1
  - Used for addressing
- Two Address Generators:
  - Auxiliary Register Arithmetic Unit

Rest of Registers

- DP : Data page pointer (similar to segment register)
- BK : *Block size (Described Later)*
- SP : System stack pointer
- ST : Status register
- IE : CPU/DMA interrupt enable register
- IF : CPU interrupt flag register
- IOF : I/O Flags register (control XF0 and XF1)

IF Bits

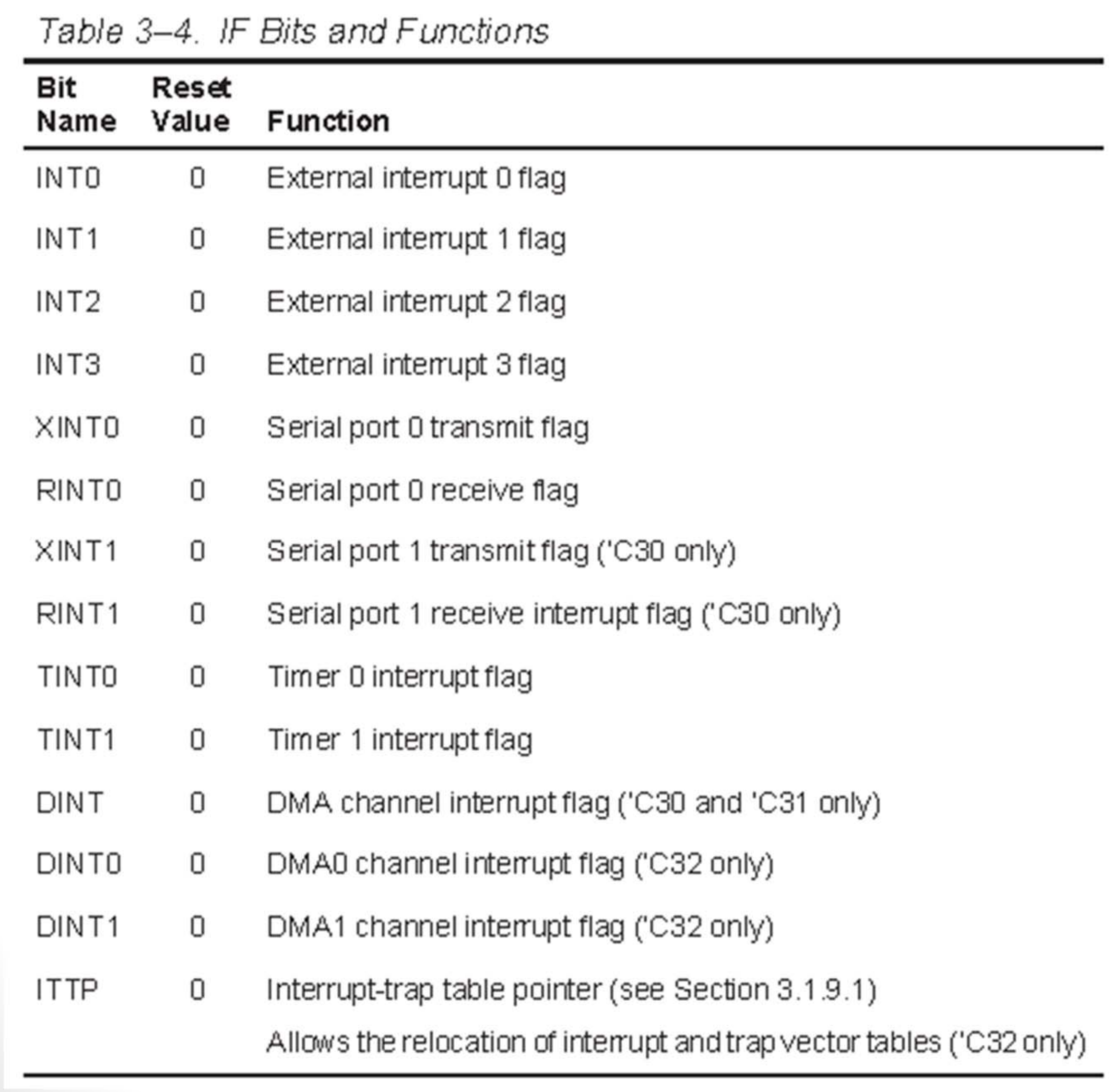


Table 3–4. IF Bits and Functions

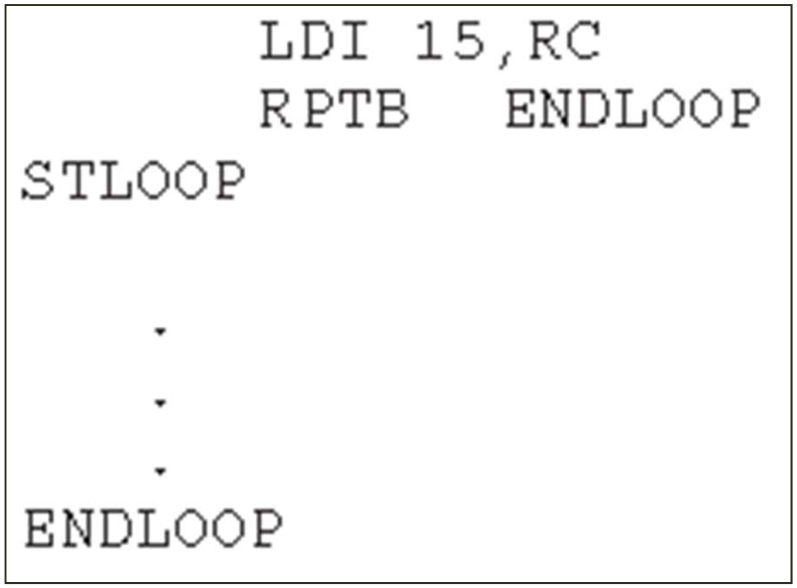
Bit Name	Reset Value	Function
INT0	0	External interrupt 0 flag
INT1	0	External interrupt 1 flag
INT2	0	External interrupt 2 flag
INT3	0	External interrupt 3 flag
XINT0	0	Serial port 0 transmit flag
RINT0	0	Serial port 0 receive flag
XINT1	0	Serial port 1 transmit flag ('C30 only)
RINT1	0	Serial port 1 receive interrupt flag ('C30 only)
TINT0	0	Timer 0 interrupt flag
TINT1	0	Timer 1 interrupt flag
DINT	0	DMA channel interrupt flag ('C30 and 'C31 only)
DINT0	0	DMA0 channel interrupt flag ('C32 only)
DINT1	0	DMA1 channel interrupt flag ('C32 only)
ITTP	0	Interrupt-trap table pointer (see Section 3.1.9.1) Allows the relocation of interrupt and trap vector tables ('C32 only)

Zero-Delay Loops

- Loops: extensively used in applications
- Great amount of time is usually consumed on “Checking the Loop Condition” in each iteration
- Hardware can handle simple loops
  - No over head for checking the loop condition

Zero-Delay Loop Registers

- Repeat-Counter RC
  - RC = n
  - Causes n+1 iterations
- RS : Repeat Start Address
- RE : Repeat End Address



```
        LDI 15,RC
        RPTB   ENDLOOP
STLOOP


    .
    .
    .
ENDLOOP
```

Memory Map

- Page 92
- Microcomputer/Boot Loader Mode

- Page 97
- Peripheral related Memory Mapped Registers

Addressing Modes

- Register Addressing
  - Register contains the operand

```
ABSF      R1                 ; R1 = |R1|
```

- Direct Addressing
  - Instruction contains address of operand directly

```
ADDI   @0BCDEh,R7
```

- Indirect Addressing
  - Auxiliary register contains the address to operand

```
LDI         *AR0,R0
```

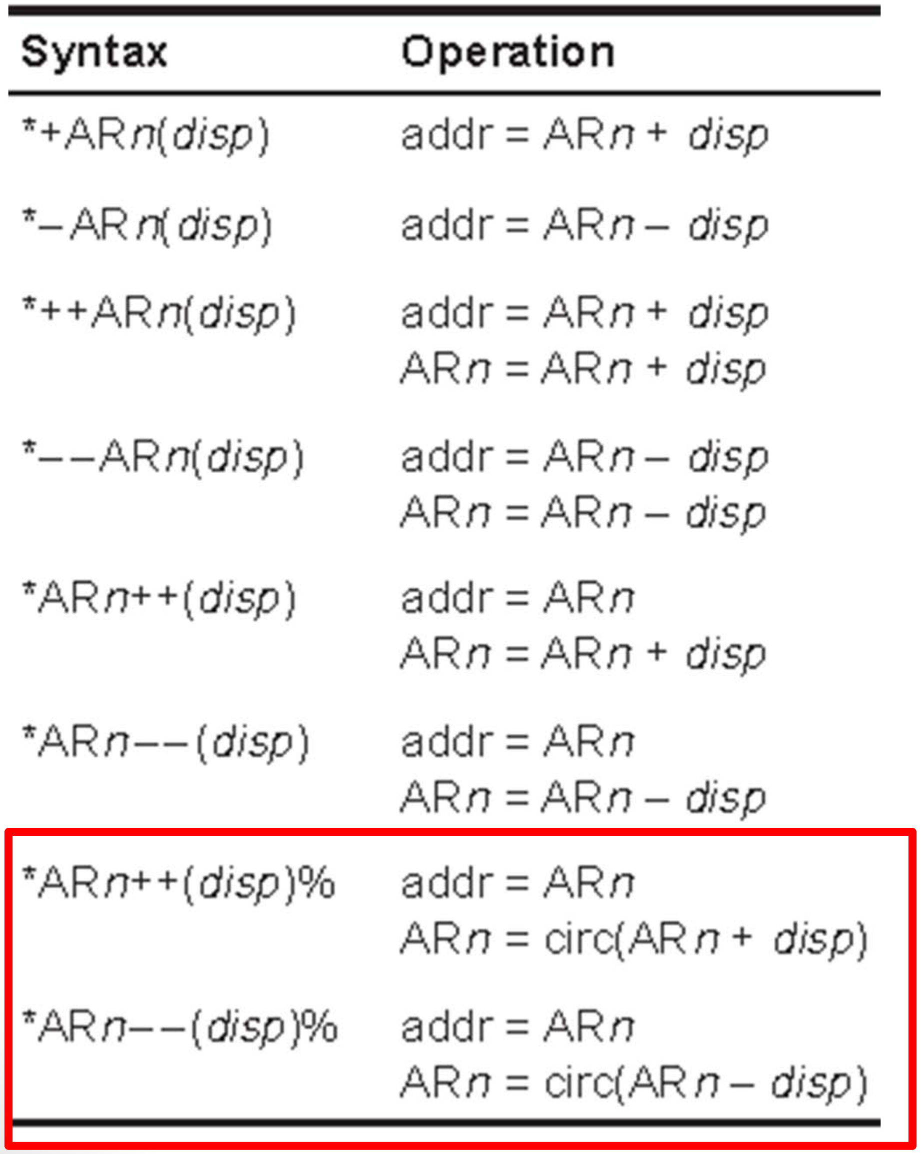
Indirect Addressing

```
MPYF   *AR2++,R1
```

```
MPYF   *++AR0(IR1),R0
```

```
LDF    *-AR3(2),R1
```

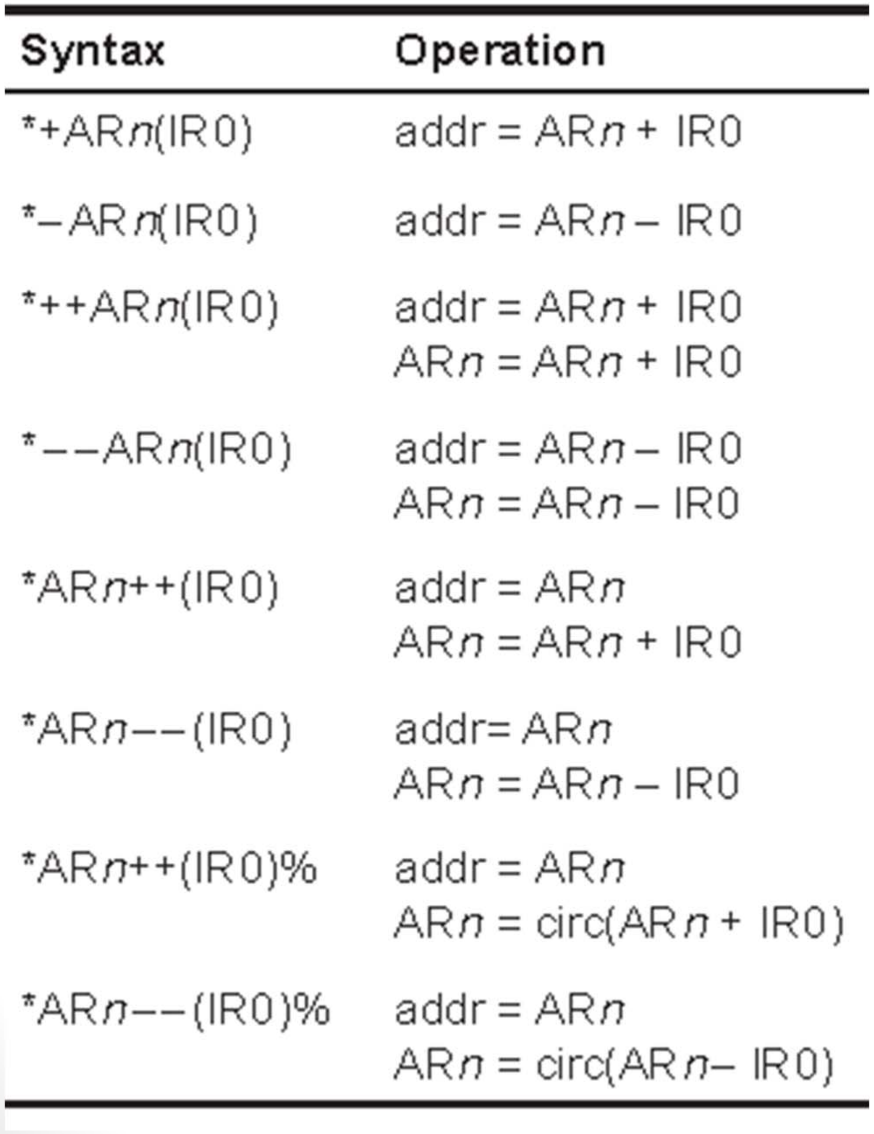
Indirect Addressing (2)



Syntax	Operation
*+AR*n*(*disp*)	addr = AR*n* + *disp*
*–AR*n*(*disp*)	addr = AR*n* – *disp*
*++AR*n*(*disp*)	addr = AR*n* + *disp* AR*n* = AR*n* + *disp*
*––AR*n*(*disp*)	addr = AR*n* – *disp* AR*n* = AR*n* – *disp*
*AR*n*++(*disp*)	addr = AR*n* AR*n* = AR*n* + *disp*
*AR*n*––(*disp*)	addr = AR*n* AR*n* = AR*n* – *disp*
*AR*n*++(*disp*)%	addr = AR*n* AR*n* = circ(AR*n* + *disp*)
*AR*n*––(*disp*)%	addr = AR*n* AR*n* = circ(AR*n* – *disp*)

Circular Addressing

Indirect Addressing (3)



Syntax	Operation
*+ARn(IR0)	addr = ARn + IR0
*–ARn(IR0)	addr = ARn – IR0

*++ARn(IR0)	addr = ARn + IR0
	ARn = ARn + IR0

*– –ARn(IR0)	addr = ARn – IR0
	ARn = ARn – IR0

*ARn++(IR0)	addr = ARn
	ARn = ARn + IR0

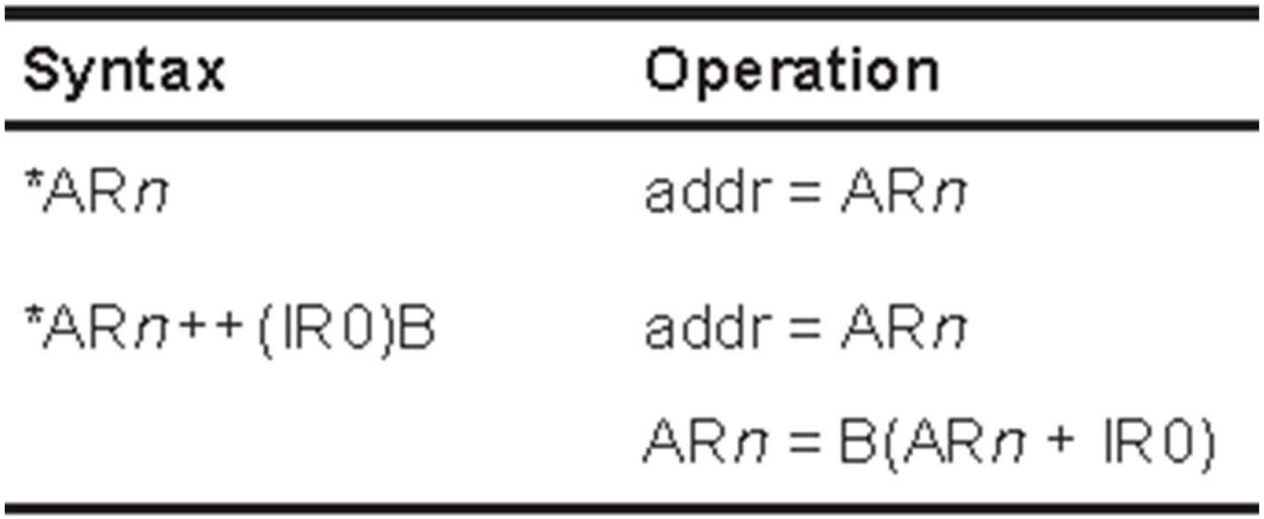
*ARn– –(IR0)	addr= ARn
	ARn = ARn – IR0

*ARn++(IR0)%	addr = ARn
	ARn = circ(ARn + IR0)

*ARn– –(IR0)%	addr = ARn
	ARn = circ(ARn– IR0)

Indirect Addressing (4)

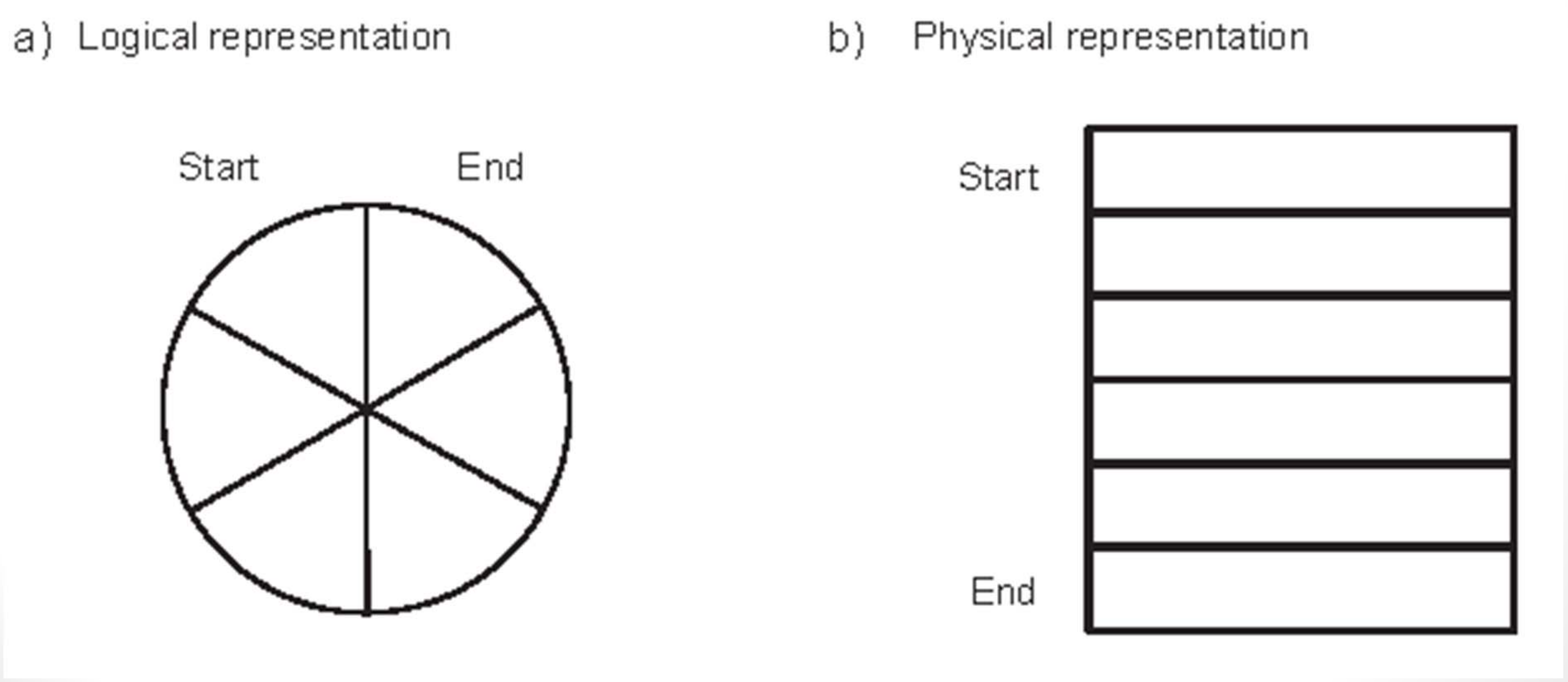
- Bit reversed addressing



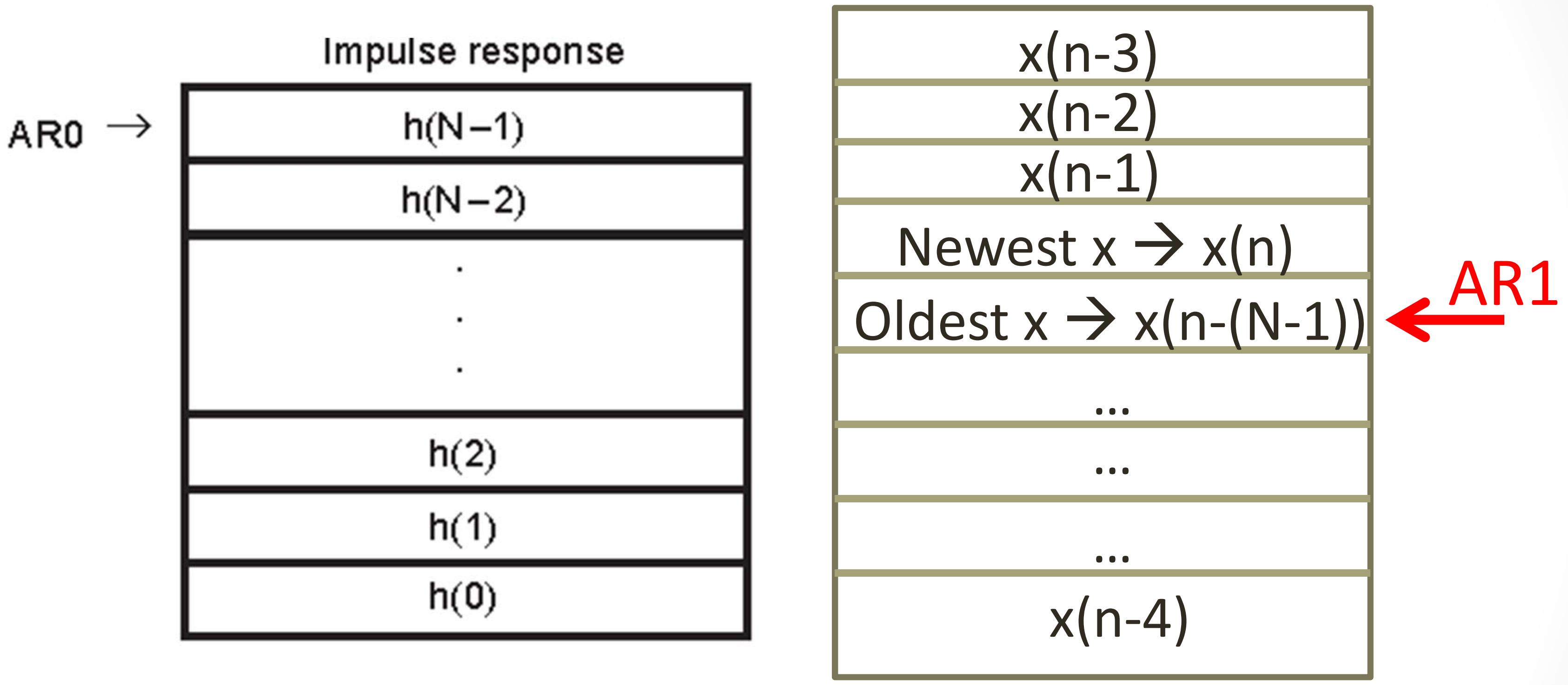
Syntax	Operation
*ARn	addr = ARn
*ARn++(IR0)B	addr = ARn
	ARn = B(ARn + IR0)

Circular Addressing

- Very useful in DSP algorithms
- BK Register (Block Size Register)
  - Holds the size of circular buffer



FIR Filter Implementation



$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

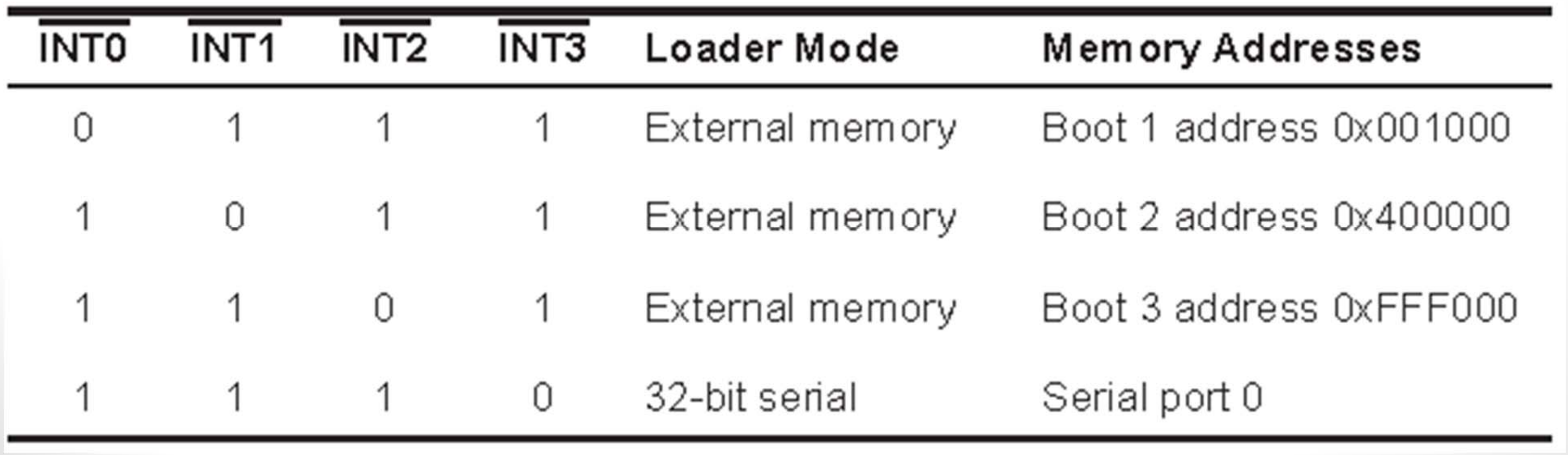
FIR Filter Main Loop

```
TOP     LDF     IN,R3
        STF     R3,*AR1++%

        LDF     0,R0
        LDF     0,R2
*
*       Filter
*
        RPTS    N-1
        MPYF3   *AR0++%,*AR1++%,R0
||      ADDF3   R0,R2,R2
        ADDF    R0,R2
*
        STF     R2,Y
        B       TOP
```

Boot Loader

- A Program Written and Stored on C31 & C32 memories
- Responsible for:
  - Receiving the main application from EPROM, serial port, …
  - And copying it into local memory
  - And Executing it right then
- INT pins indicate
  - What boot loader should do

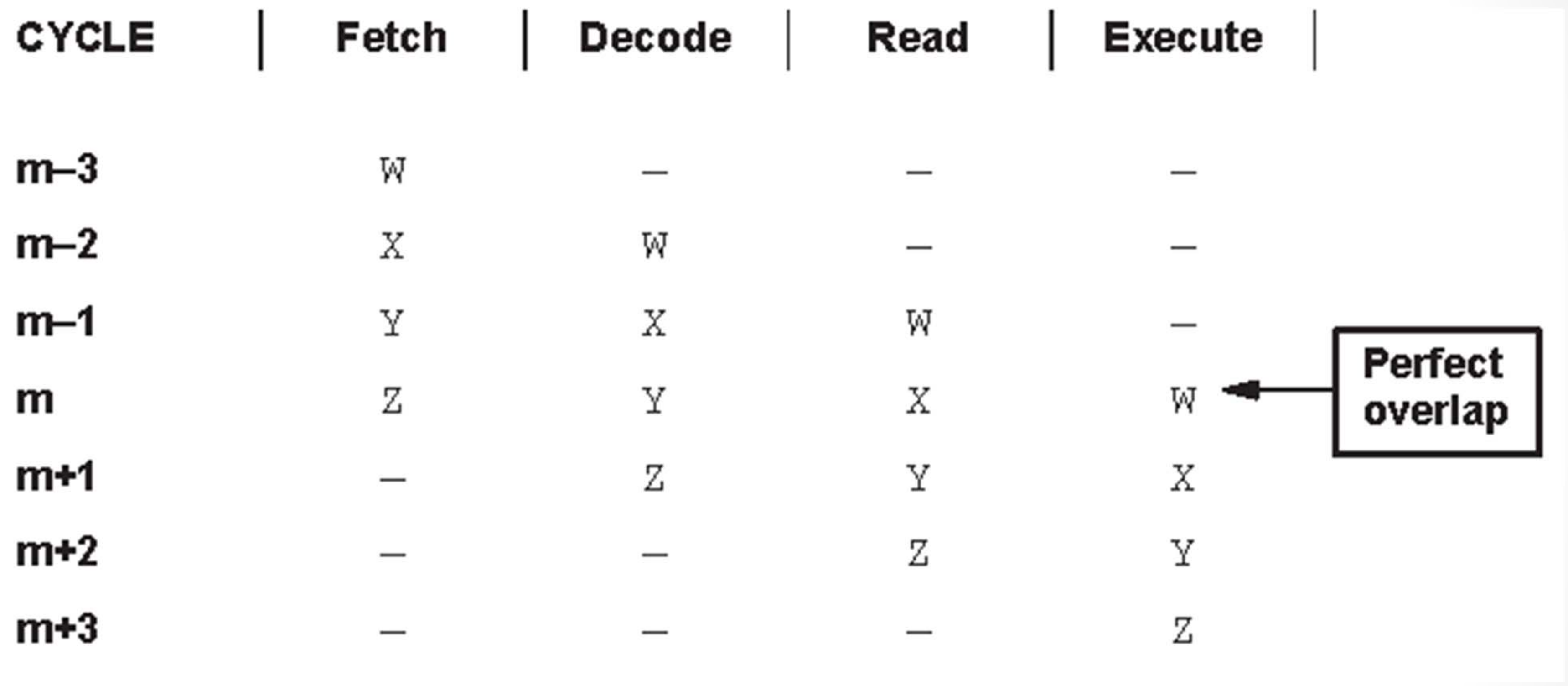


$\overline{INT0}$	$\overline{INT1}$	$\overline{INT2}$	$\overline{INT3}$	Loader Mode	Memory Addresses
0	1	1	1	External memory	Boot 1 address 0x001000
1	0	1	1	External memory	Boot 2 address 0x400000
1	1	0	1	External memory	Boot 3 address 0xFFF000
1	1	1	0	32-bit serial	Serial port 0

Branch Operations

- Three categories of branches:
  - Standard Branch
    - Empty the pipeline before performing the branch
  - Delayed Branch
    - Do not empty the pipeline, execute the next three instructions
  - Conditional Delayed Branch
    - Use the conditions that exist at the end of the instruction before the branch
    - They do not depend on instructions following the branch
    - Condition flags are set when
      - R0-R7 change
      - CMPF, CMPI, TSTB executed

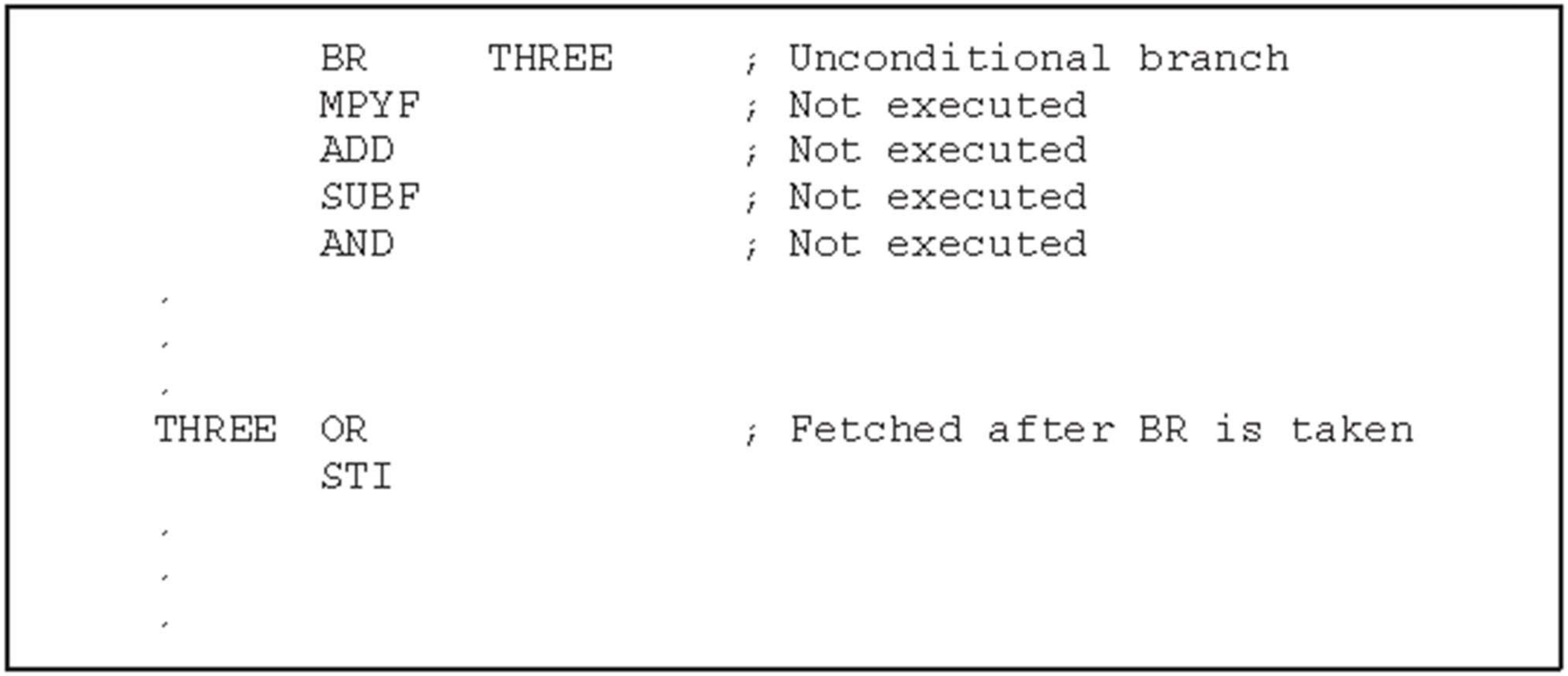
Pipeline



CYCLE	Fetch	Decode	Read	Execute
m–3	W	–	–	–
m–2	X	W	–	–
m–1	Y	X	W	–
m	Z	Y	X	W
m+1	–	Z	Y	X
m+2	–	–	Z	Y
m+3	–	–	–	Z

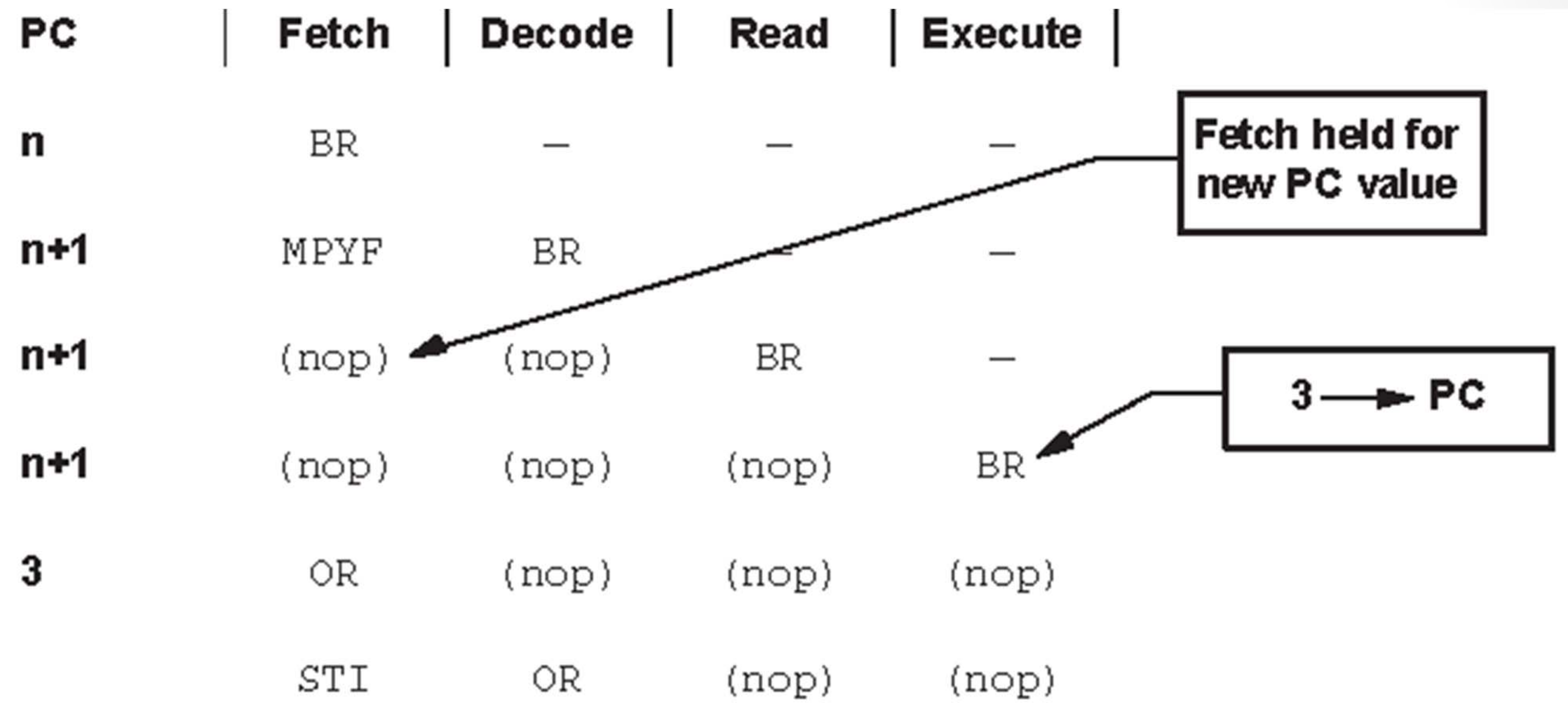
Perfect overlap

Example Standard Branch



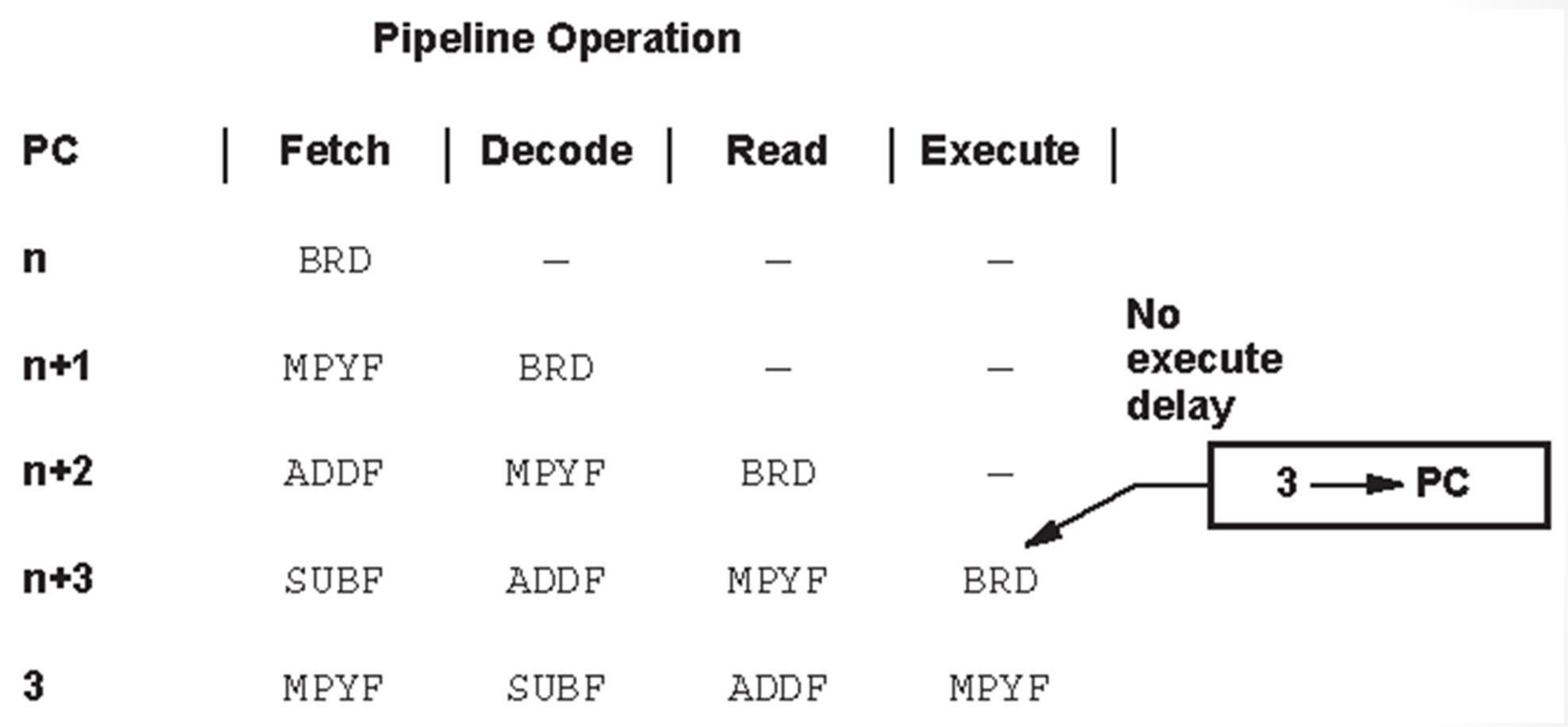
```
        BR      THREE        ; Unconditional branch
        MPYF                 ; Not executed
        ADD                  ; Not executed
        SUBF                 ; Not executed
        AND                  ; Not executed
  .
  .
  .
THREE   OR                   ; Fetched after BR is taken
        STI
  .
  .
  .
```

Example Standard Branch (2)


Fetch held for new PC value
$3 \longrightarrow$ PC

PC	Fetch	Decode	Read	Execute
n	BR	–	–	–
n+1	MPYF	BR	–	–
n+1	(nop)	(nop)	BR	–
n+1	(nop)	(nop)	(nop)	BR
3	OR	(nop)	(nop)	(nop)
	STI	OR	(nop)	(nop)

Delayed Branch

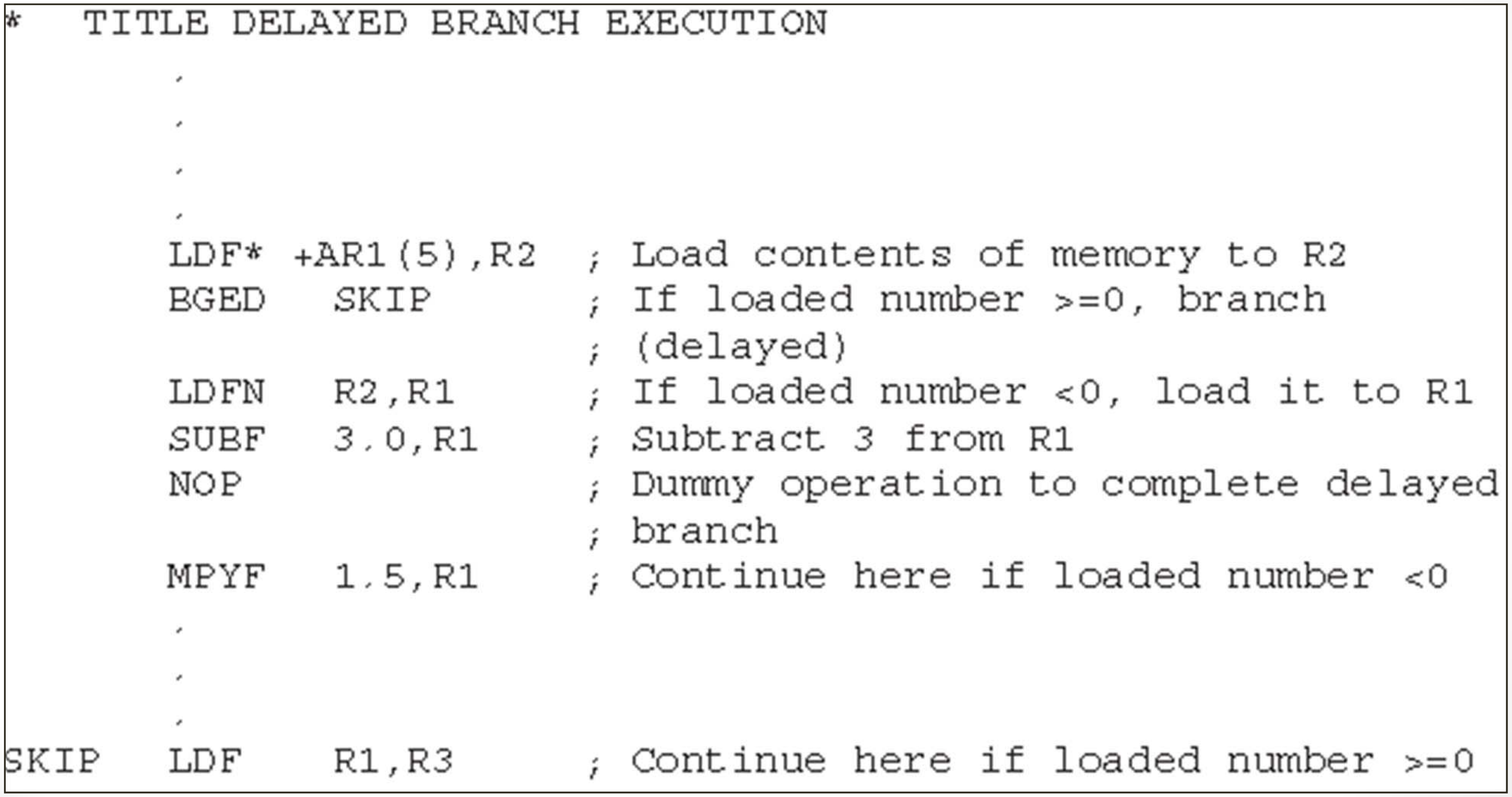


Pipeline Operation

PC	Fetch	Decode	Read	Execute
n	BRD	–	–	–
n+1	MPYF	BRD	–	–
n+2	ADDF	MPYF	BRD	–
n+3	SUBF	ADDF	MPYF	BRD
3	MPYF	SUBF	ADDF	MPYF

No execute delay
$3 \rightarrow PC$

Delayed Branch Example



```
*   TITLE DELAYED BRANCH EXECUTION
        .
        .
        .
        .
        LDF* +AR1(5),R2  ; Load contents of memory to R2
        BGED   SKIP      ; If loaded number >=0, branch
                         ; (delayed)
        LDFN   R2,R1     ; If loaded number <0, load it to R1
        SUBF   3.0,R1    ; Subtract 3 from R1
        NOP              ; Dummy operation to complete delayed
                         ; branch
        MPYF   1.5,R1    ; Continue here if loaded number <0
        .
        .
        .
SKIP    LDF    R1,R3     ; Continue here if loaded number >=0
```