MA4832

Basics of ARM Memory

Xie Ming, PhD (France)

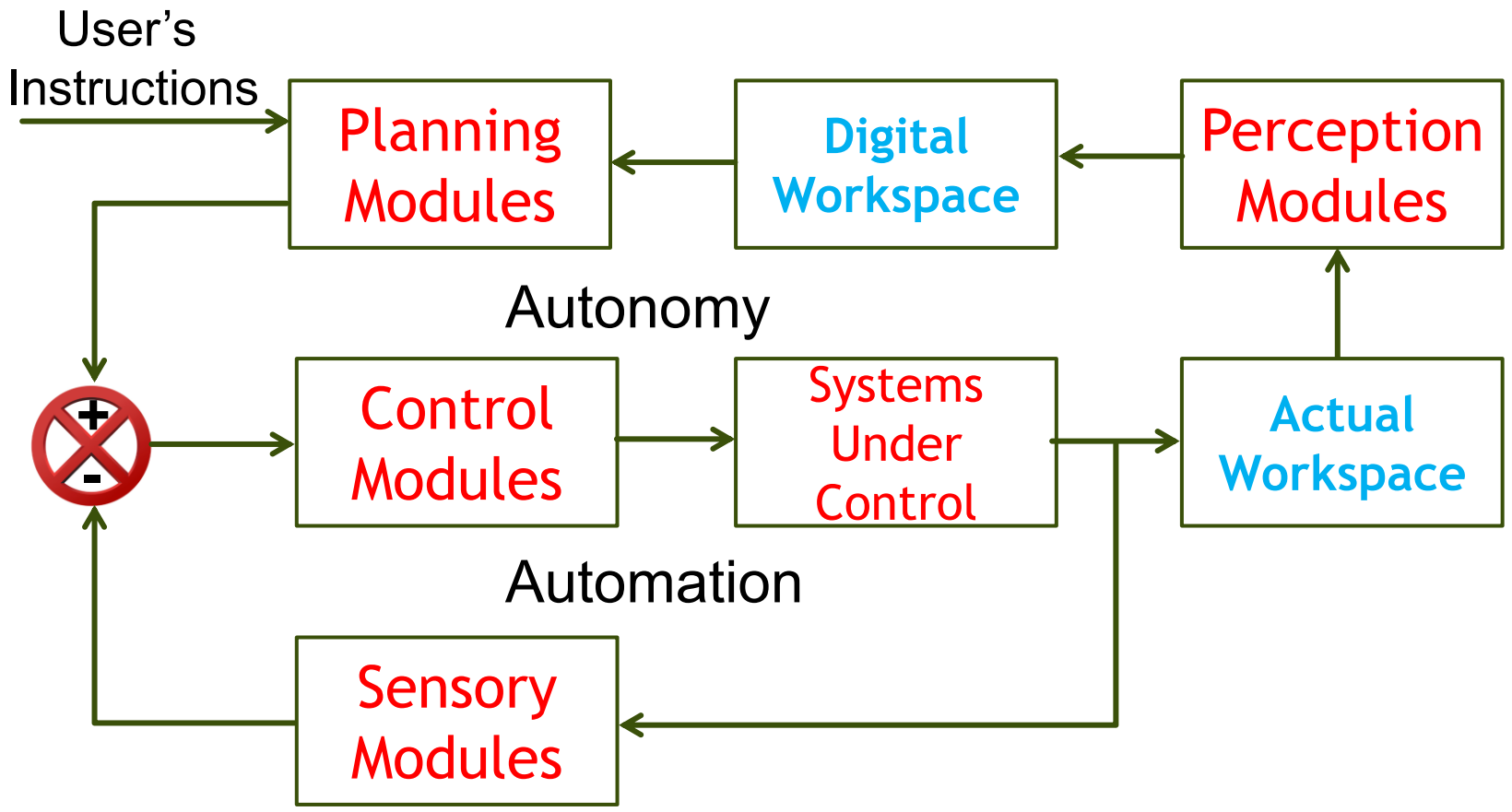
http://personal.ntu.edu.sg/mmxie

Remember your mission as MAE graduates …

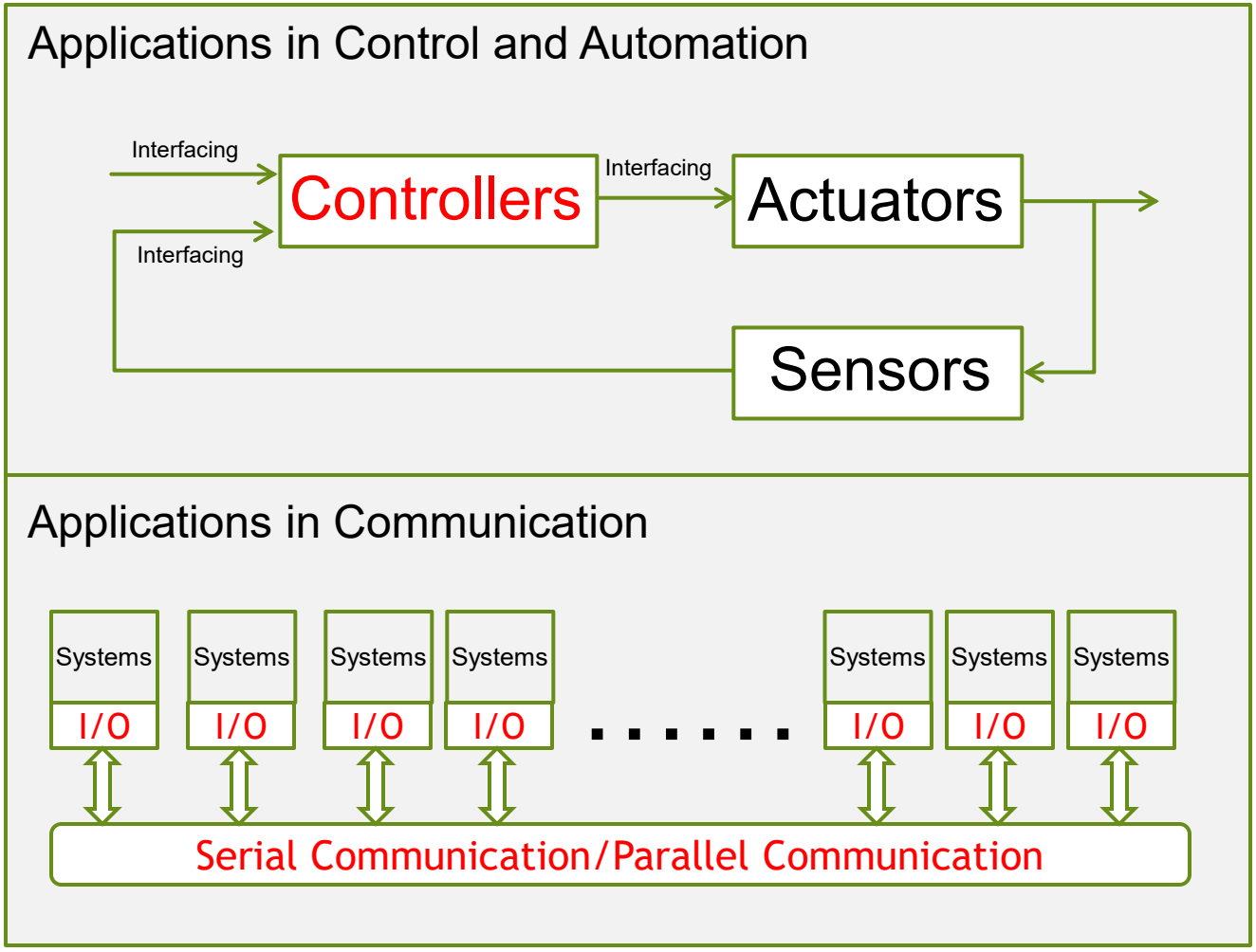
- You are here to grow your knowledge and skills so as to be able to design machines with controllable behaviors and hopefully in some intelligent ways.

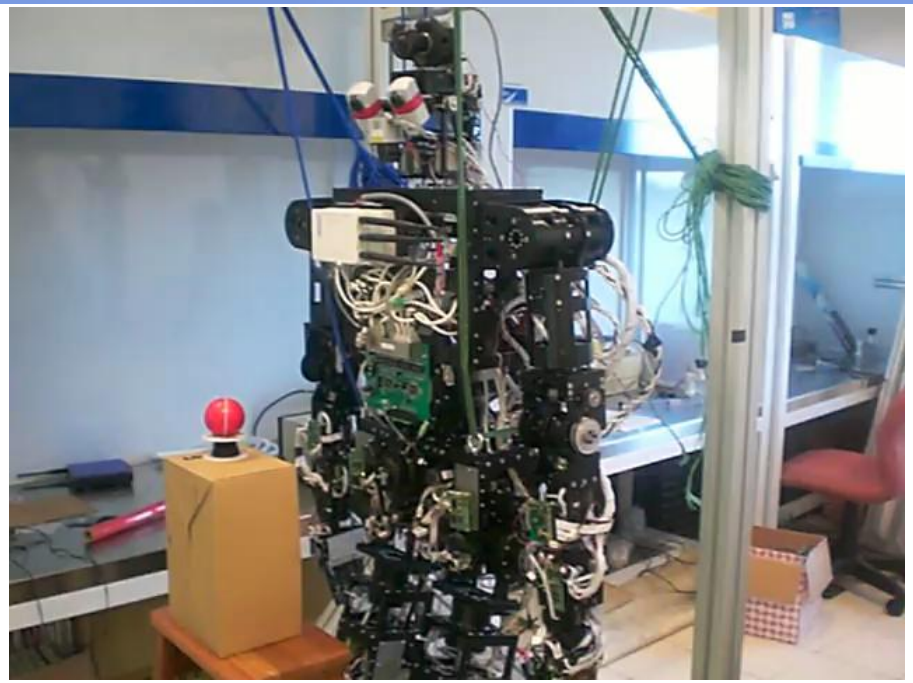
How to fulfill your mission?

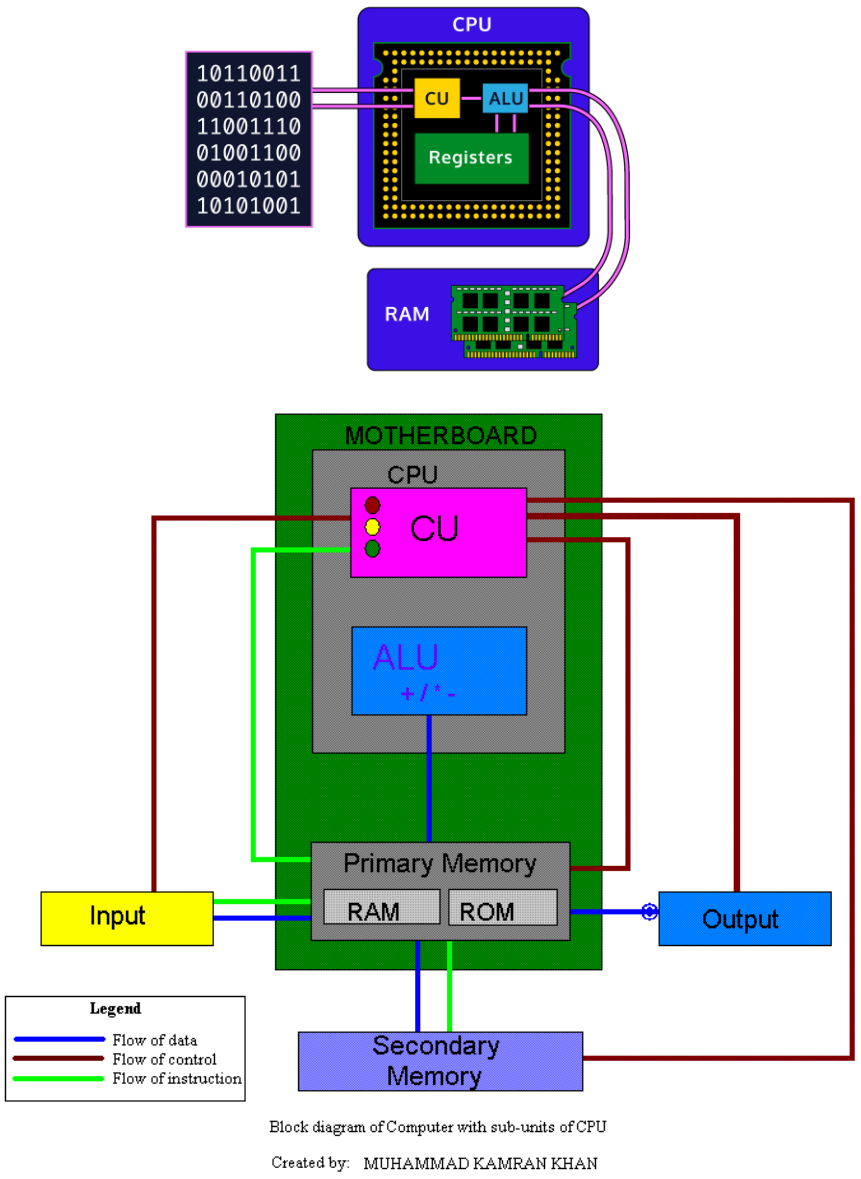
- To apply learnt knowledge and skills into the implementation of the following universal blueprint underlying all the intelligent machines or systems.



Why to study?


Applications in Control and Automation
Interfacing
Interfacing
Controllers
Interfacing
Actuators
Sensors
Applications in Communication
Systems
I/O
Systems
I/O
Systems
I/O
Systems
I/O
Systems
I/O
Systems
I/O
Systems
I/O
Serial Communication/Parallel Communication



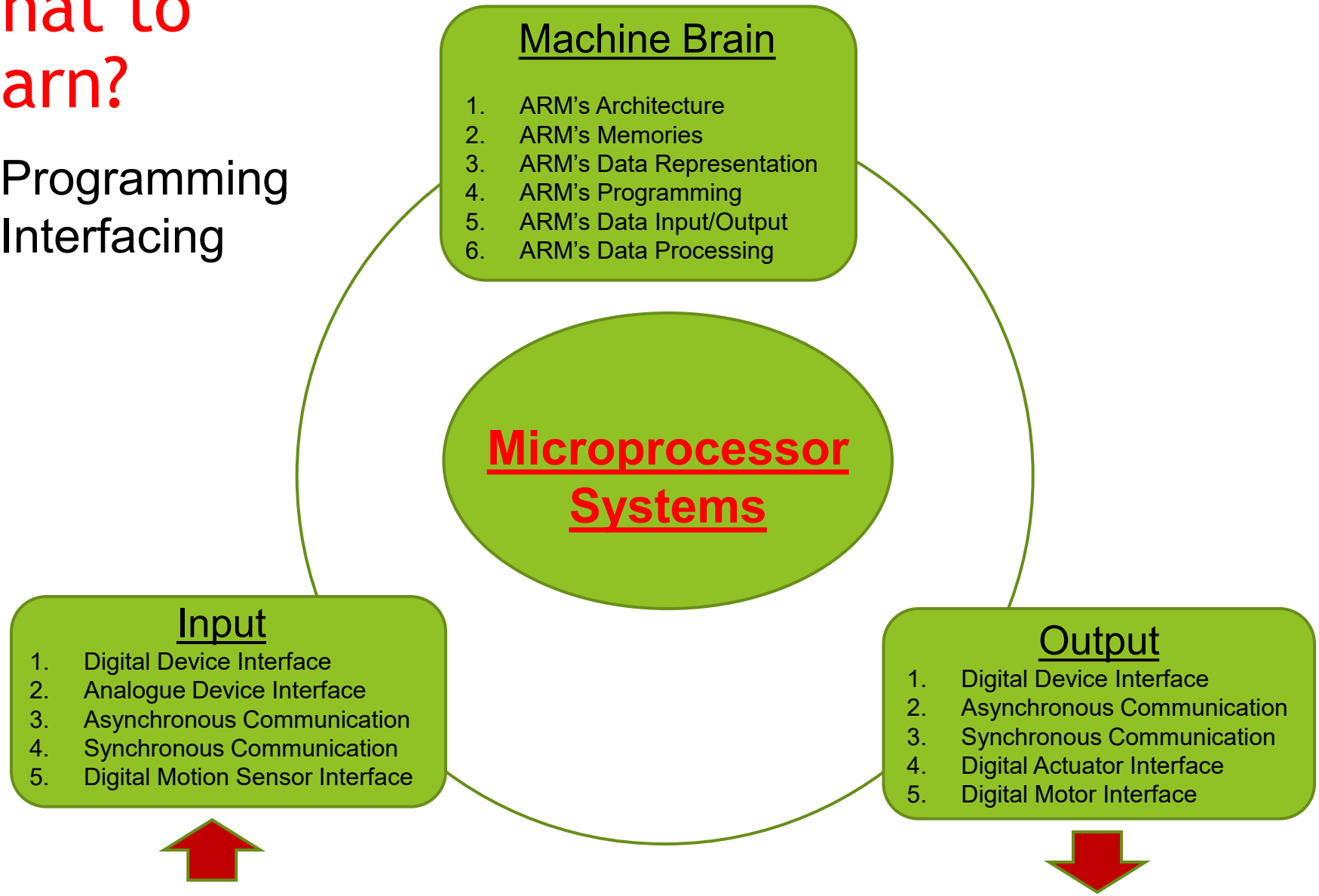


Block diagram of Computer with sub-units of CPU

Created by: MUHAMMAD KAMRAN KHAN

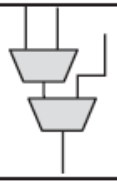
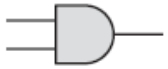
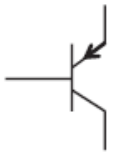
What to learn?

- Programming
- Interfacing



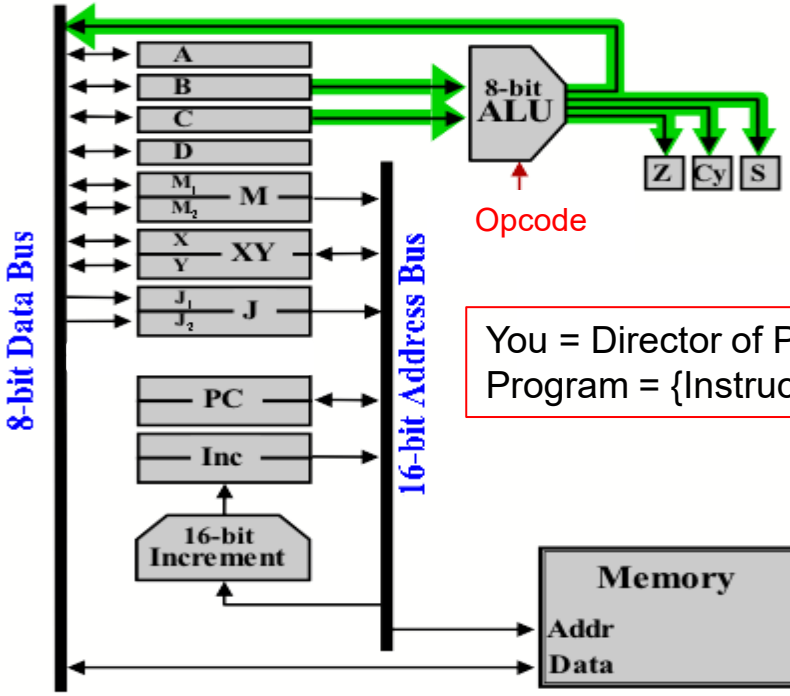
What is your role?

- Data = {Values, Symbols, Addresses, Instructions}
- Instructions = {Op Code + Addresses + Value/Symbol}

Applications/OS	
Languages	C++, Java
ISA	EOR r3,r2,r1 BEQ Table
Microarchitecture	
Gates	
Transistors	

Algorithms or Solutions

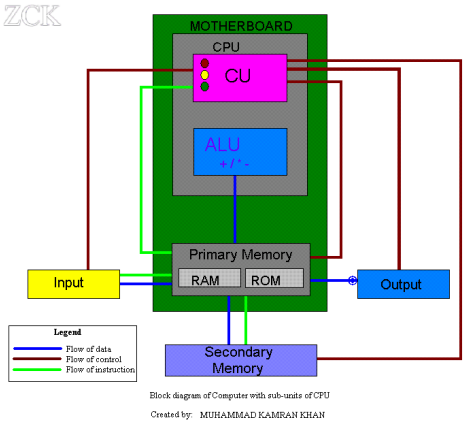
Problems to be solved



You = Director of Program
Program = {Instructions}

Memory = {Address + Data/Instruction}

How to use?



- To understand data flows inside a microcontroller.
- To translate your solutions into data flows.
- To pay attention to <memory address> and <memory data>.
  - Memory Address: Address Label/Name and Address Value.
  - Memory Data: Data Label/Name and Data Value.

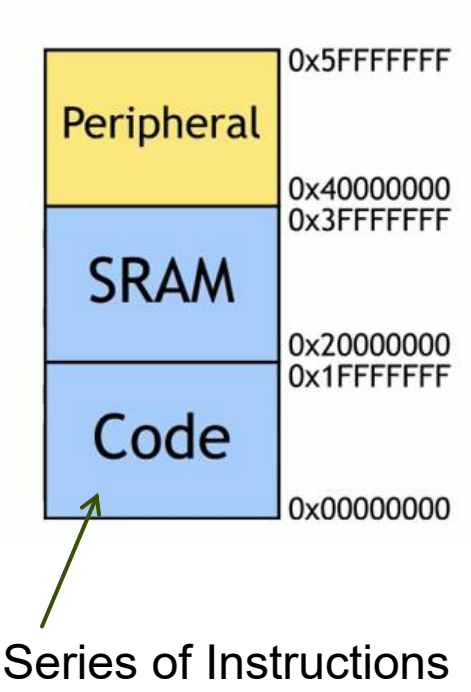
Memory Content

Data	Memory Address

- To pay attention to <memory address> and <memory code>.
  - Memory Address: Address Label/Name and Address Value.
  - Memory Code: Code Label/Name and Code Value.

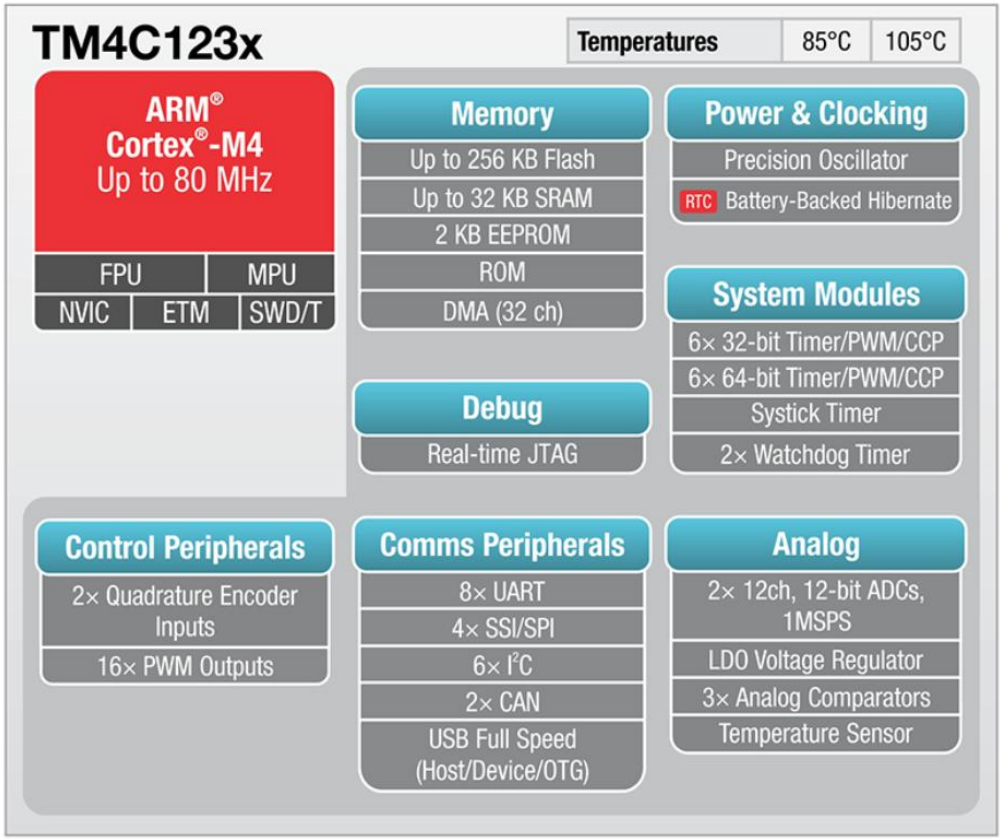
Memory Content

Instruction	Memory Address



Example of Using I/O Modules

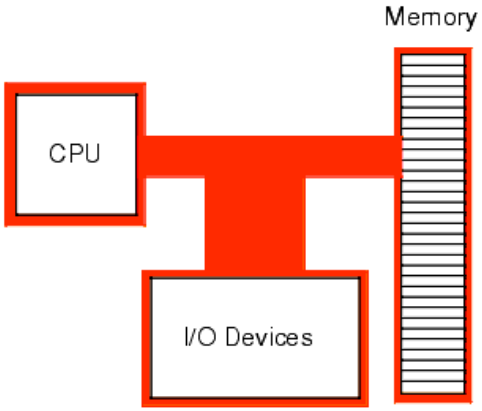
- Configure Control Registers
- Clear/Monitor Status Registers
- Read/Write Data Registers
- Instructions:
  - MOV <address of destination>, <source of value>
  - LDR <address of destination>, <source of value>
  - STR <source of value>, <address of destination>

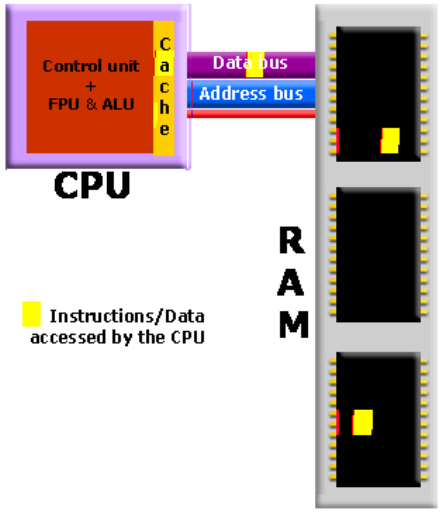



```
MOV   R0, #0x11
MOV   R1, #2560
MVN   R2, #4
MOVW  R3, #0xC0DE
MOVT  R3, #0xFEED
MOV   R4, R1
```

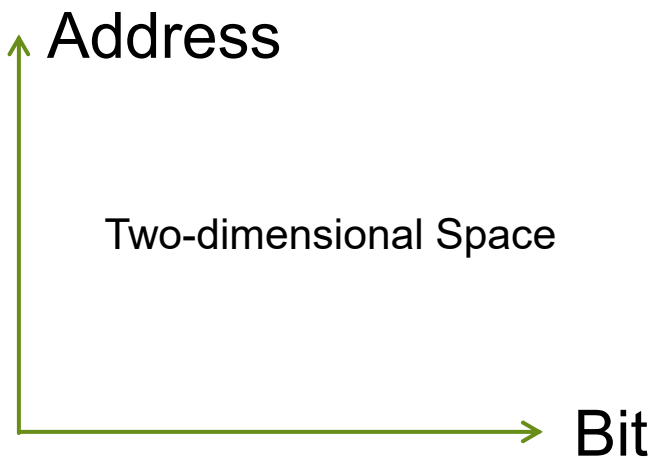
Word = 2 Bytes

Outline



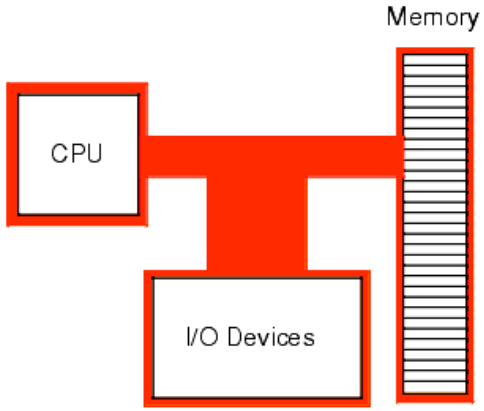


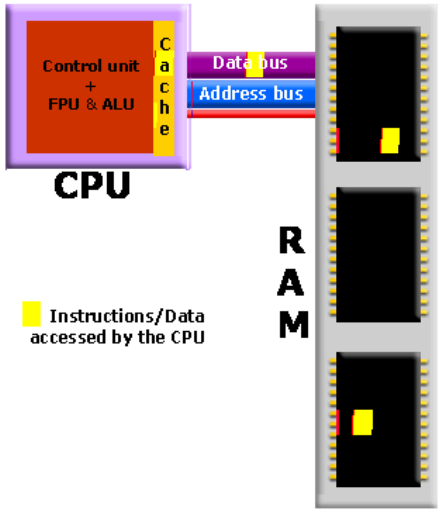
- Binary Logic Devices
- Memory Construction
- Memory Space in ARM
- Memory-Centric Operations
- Memory-Mapped I/O Devices



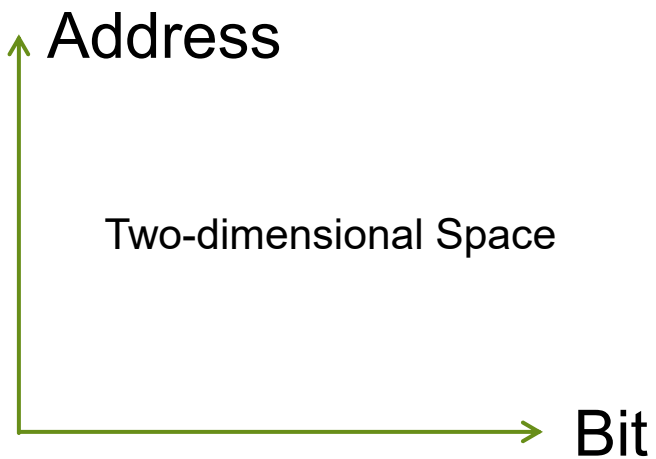
1. Memory Content
2. Memory Address
3. Memory Label

Outline





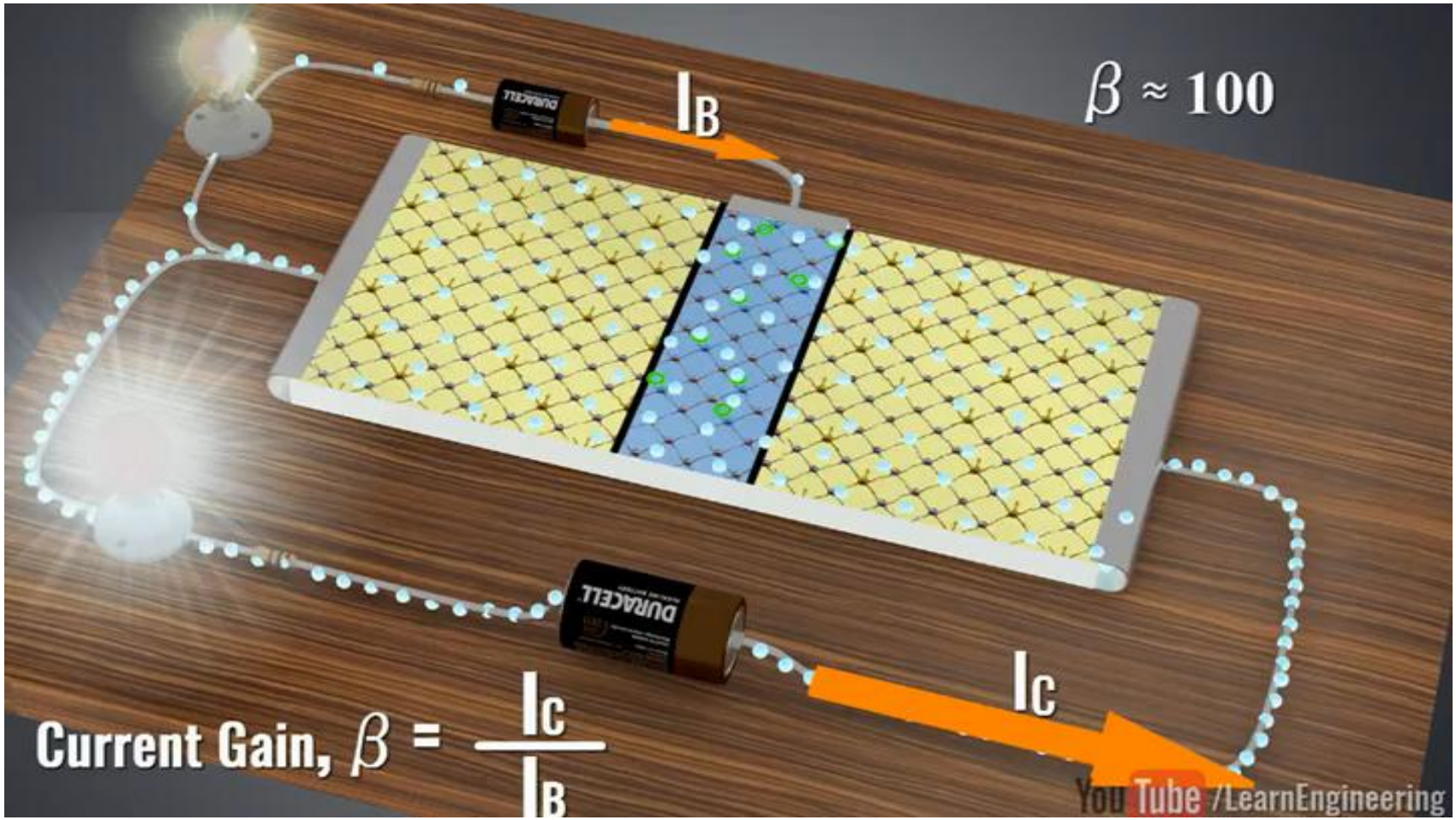
- Binary Logic Devices
- Memory Construction
- Memory Space in ARM
- Memory-Centric Operations
- Memory-Mapped I/O Devices



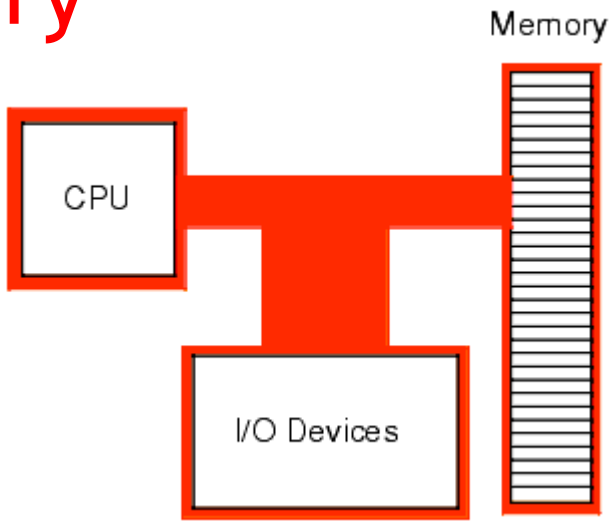
1. Memory Content
2. Memory Address
3. Memory Label

Knowledge of Transistor (1)

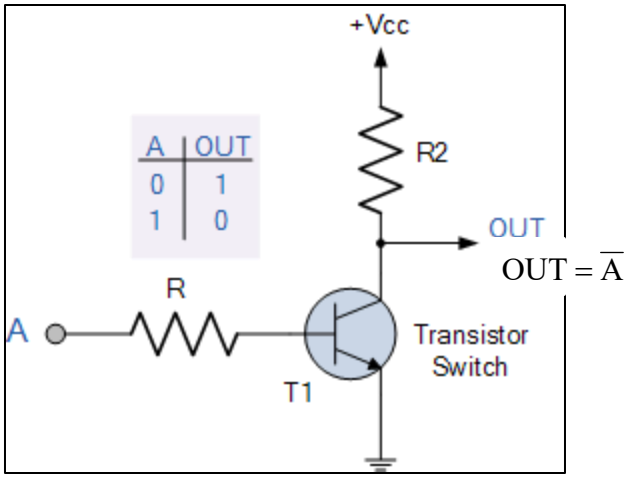
Knowledge of Transistor (2)



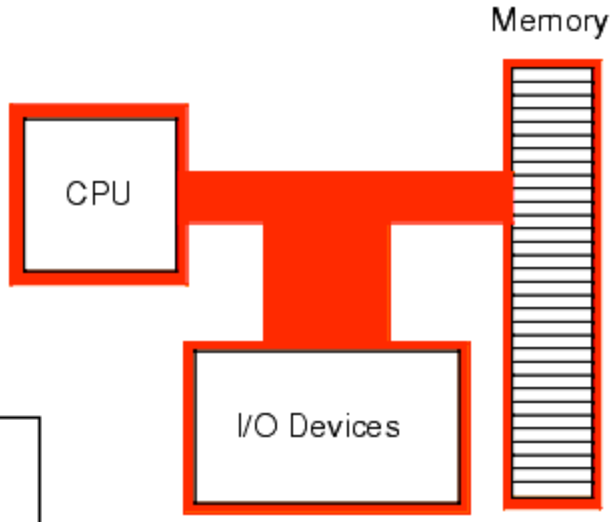
Digital computers are binary number systems



Binary Values	Implementation
1 (Logic High)	+5 V (Logic High)
0 (Logic Low)	0 V (Logic Low)

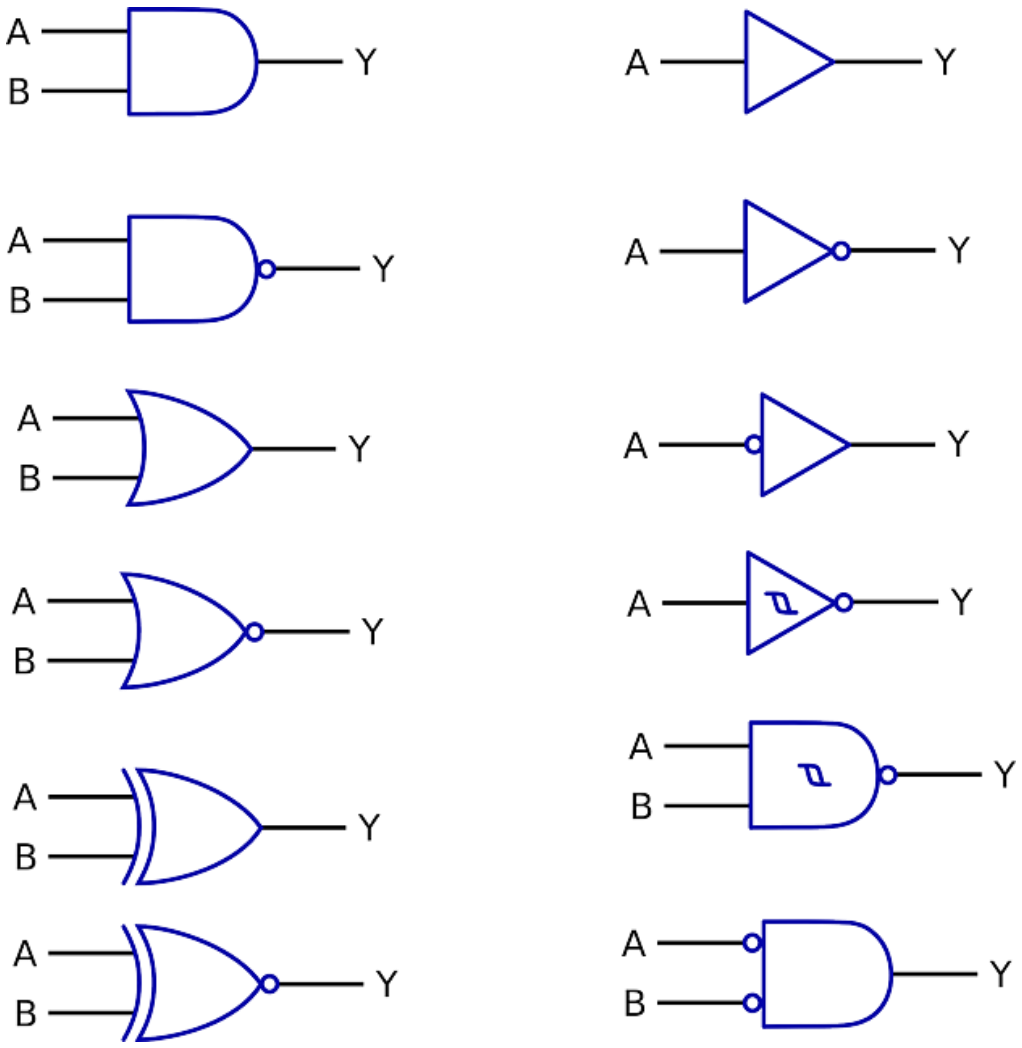


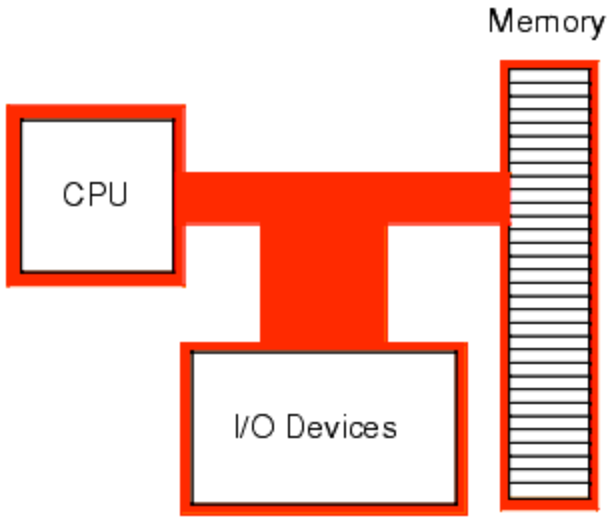
Binary Logic Operations



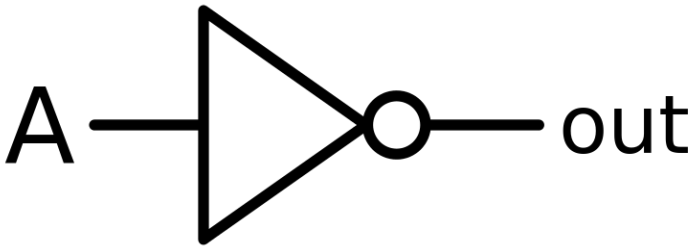
NOT	A′ ~A $\overline{A}$ ¬A
AND	AB A*B A•B A ∧ B A ∩ B
OR	A+B A ∨ B A ∪ B
NAND	(AB)′ $\overline{AB}$
NOR	(A+B)′ $\overline{A+B}$
XOR	A ⊕ B A @ B
XNOR	(A ⊕ B)′ $\overline{A \oplus B}$ (A @ B)′ $\overline{A @ B}$

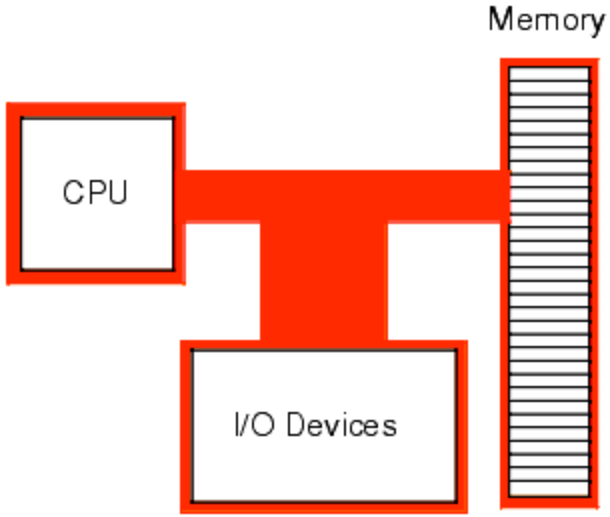
Binary Logic Devices





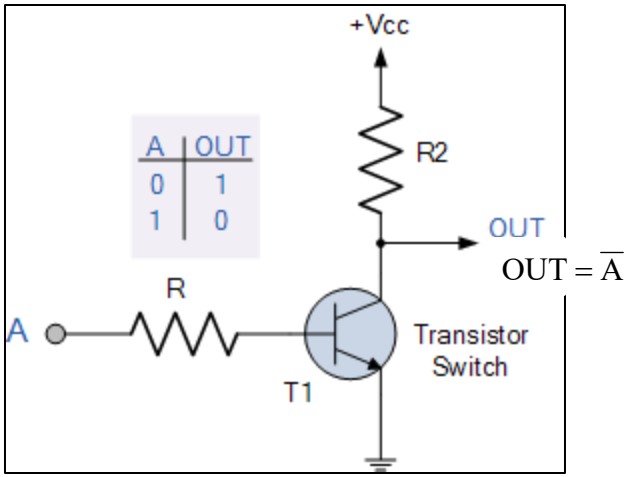
Example: NOT





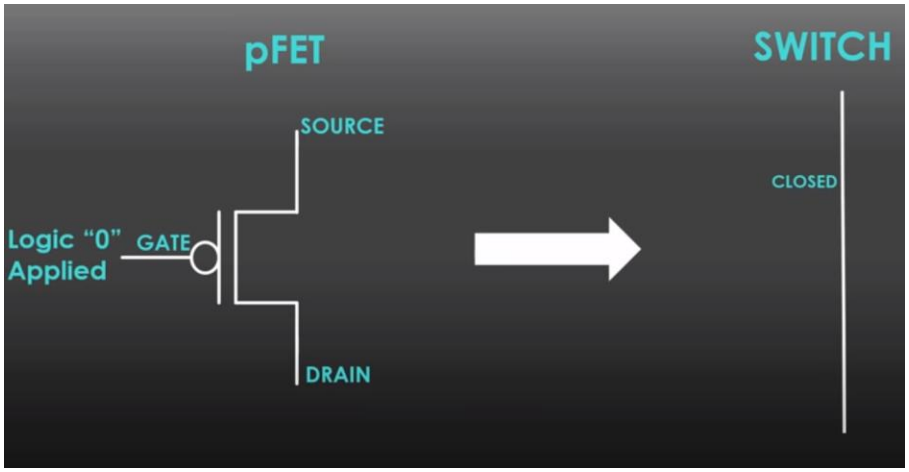
A	$\overline{A}$
1	0
0	1

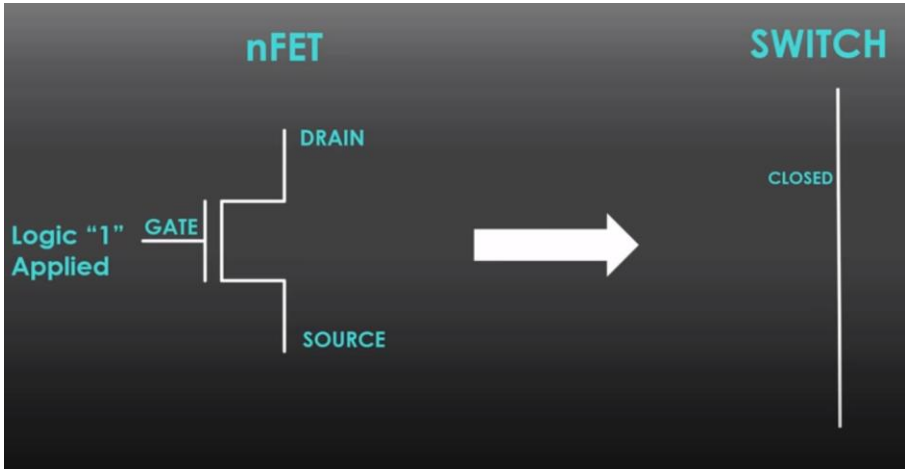
Logic Gate: Not



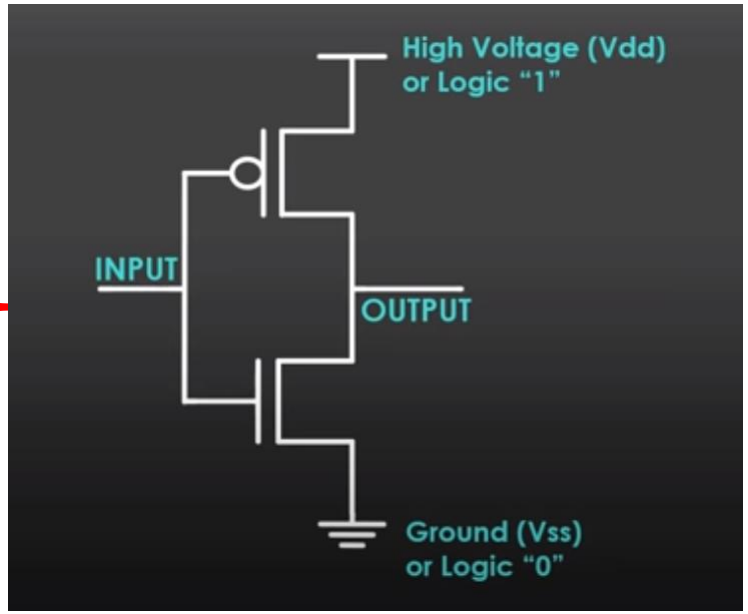
More Example

FET = Field Effect Transistor

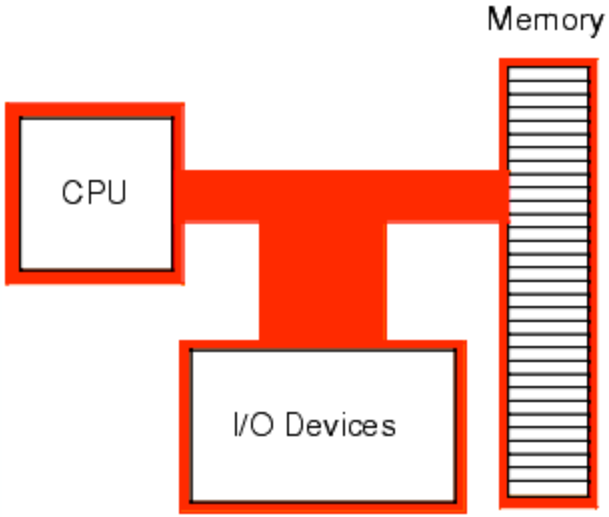




Logic Gate: Not

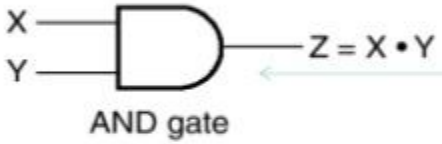


Example: AND



AND Gate

- Logic Symbol, Truth Table And Logic Expression



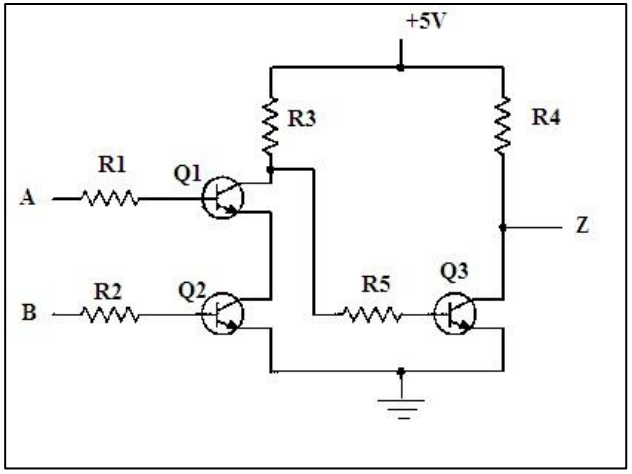
Logic Symbol

X	Y	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

Logic Expression

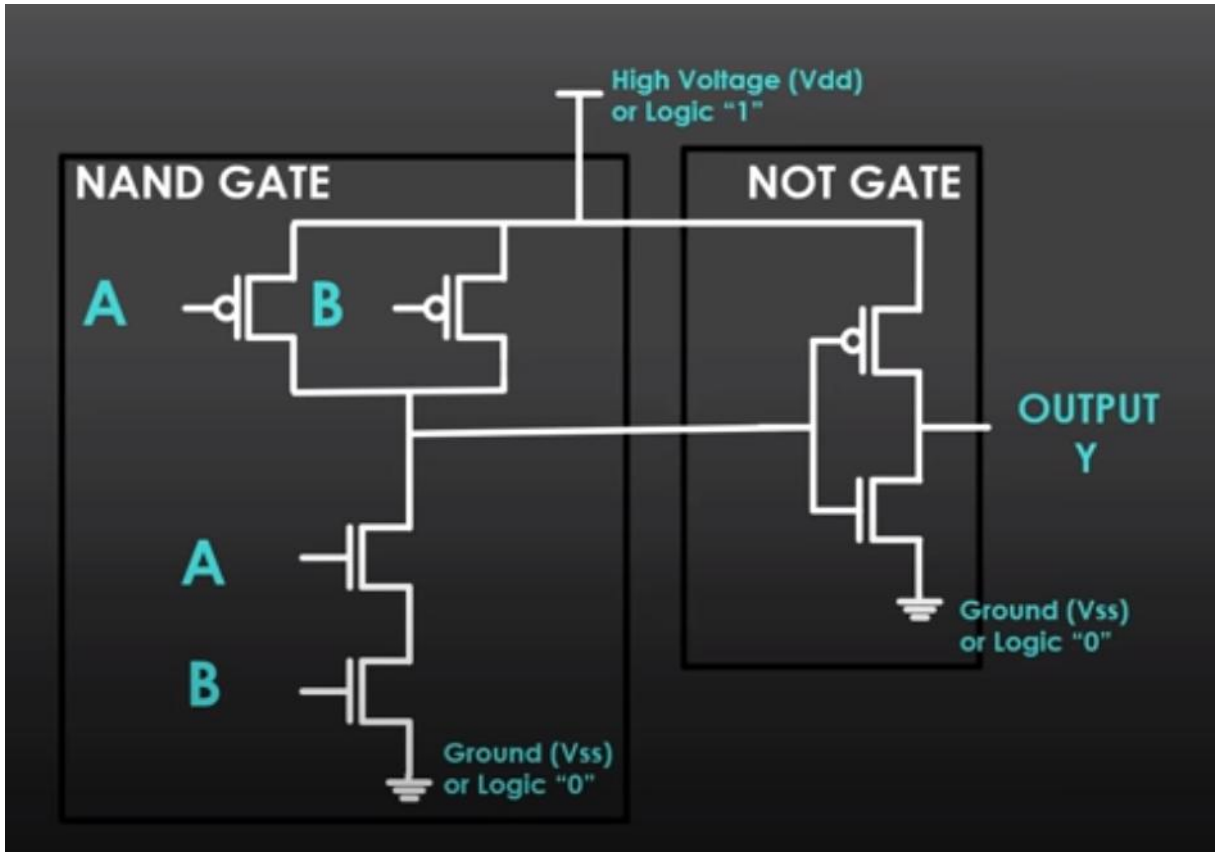
Truth Table

Logic Gate: AND



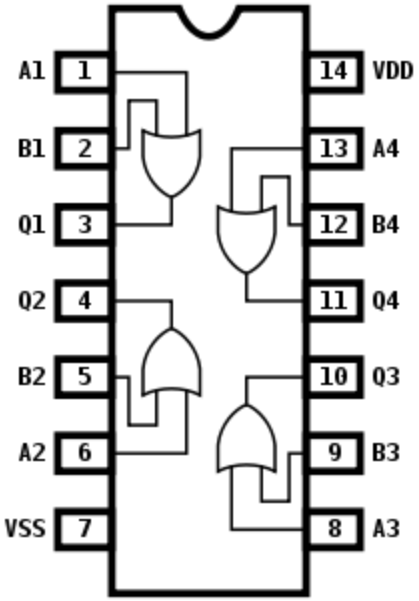
More Example

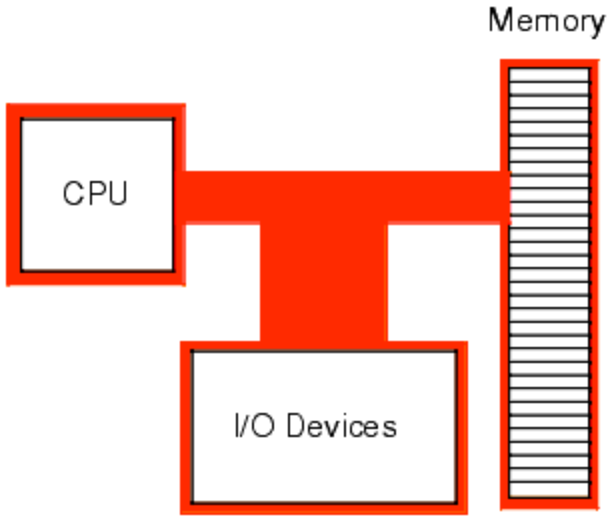
Logic Gate: AND

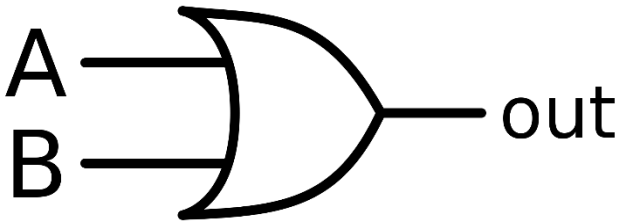


A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Example: OR

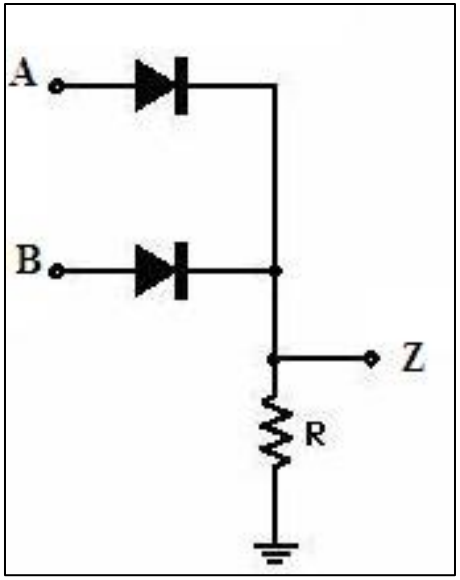




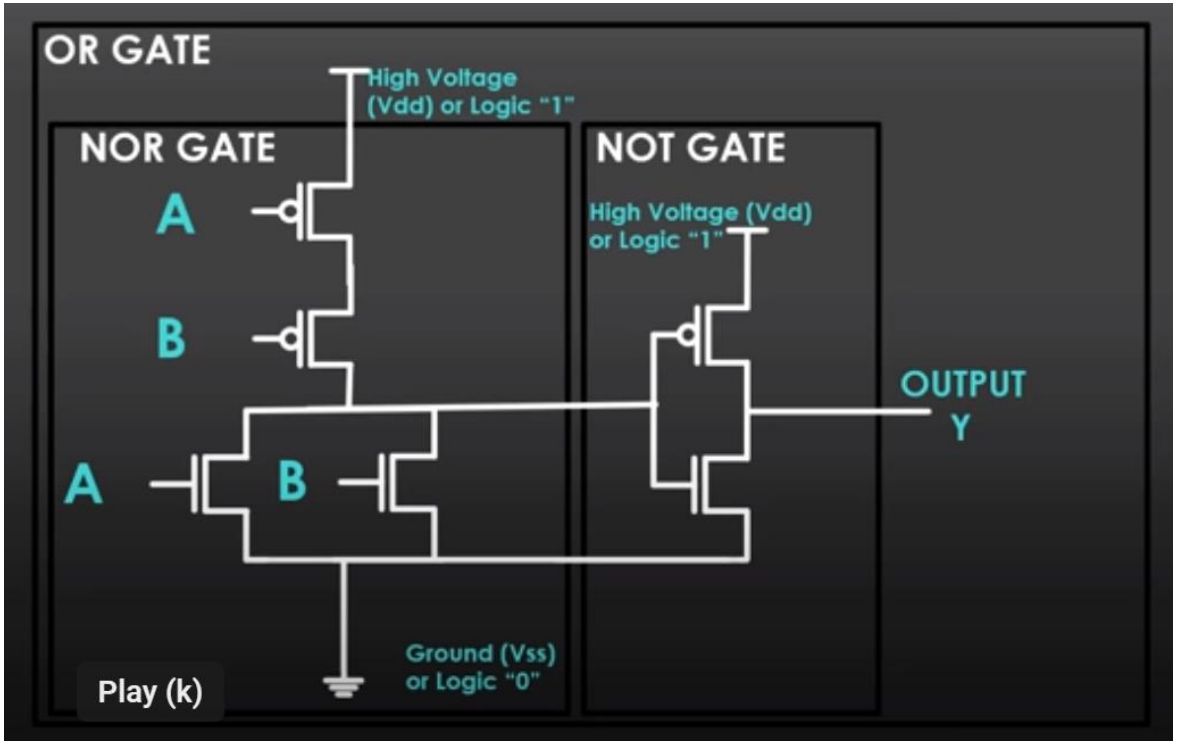


A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

Logic Gate: OR



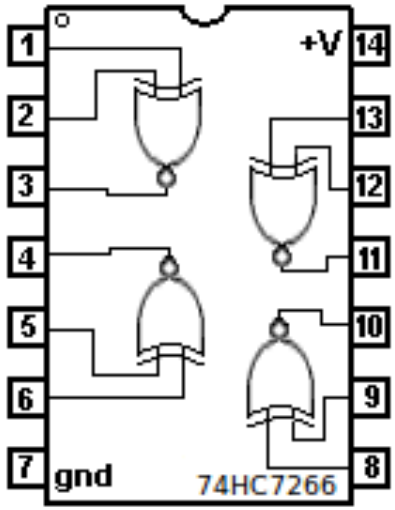
More Example

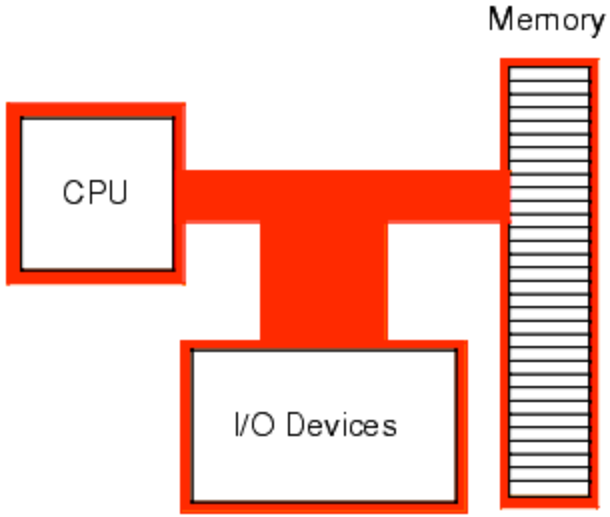


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

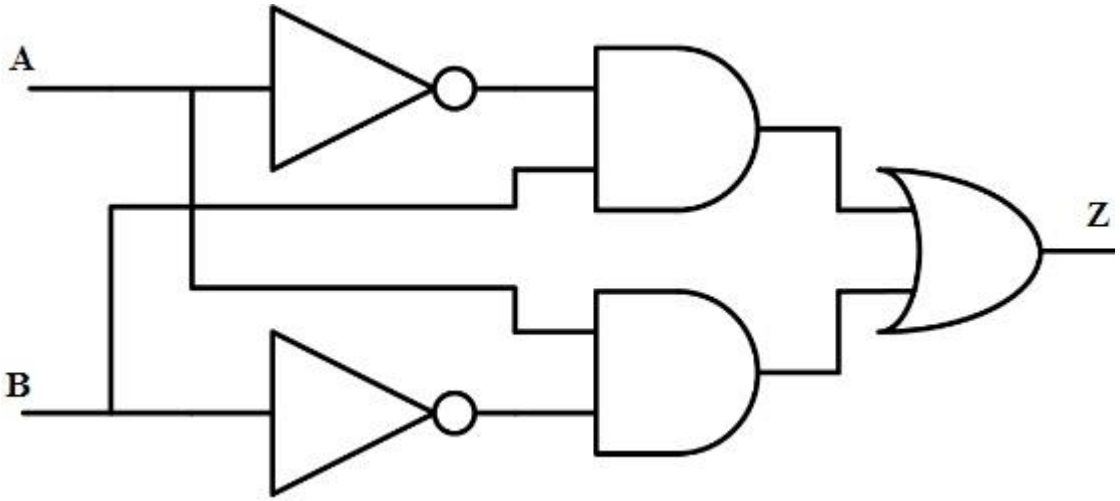
Example: XOR

Exclusive-OR gate

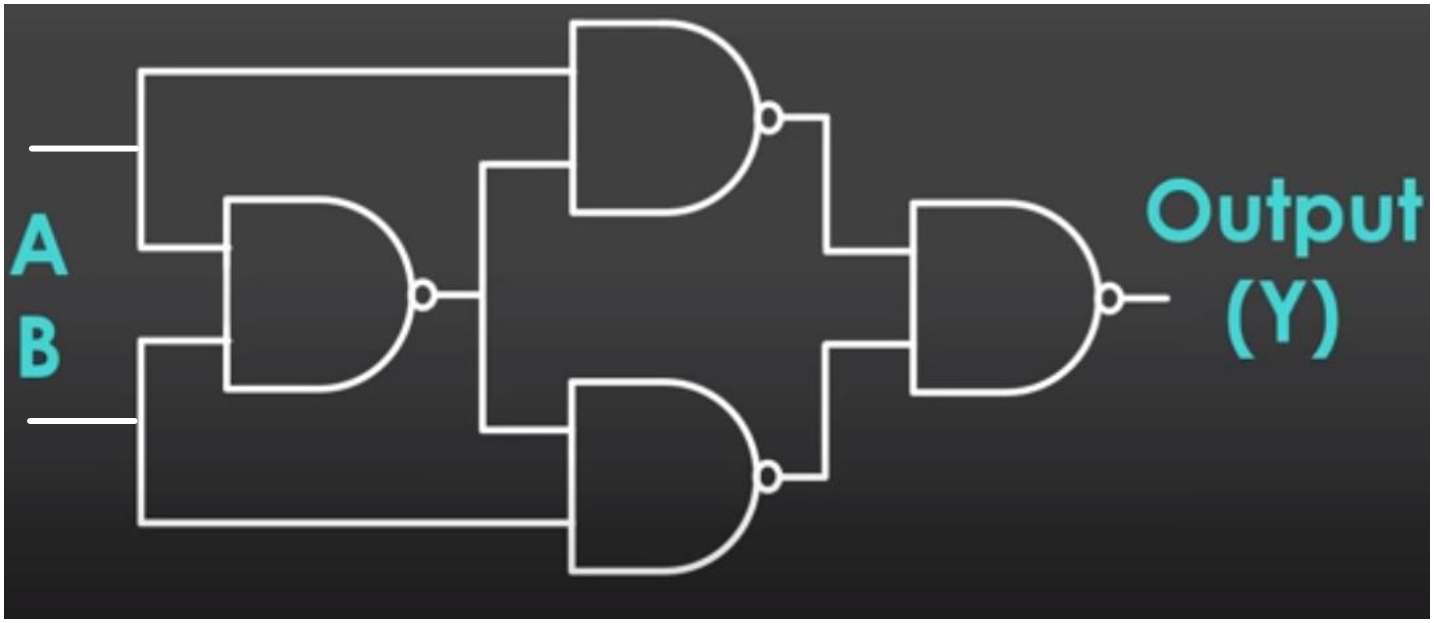




A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0



More Example



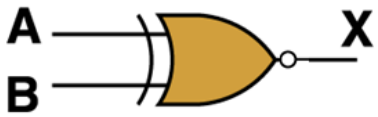
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Example: XNOR

Boolean Expression

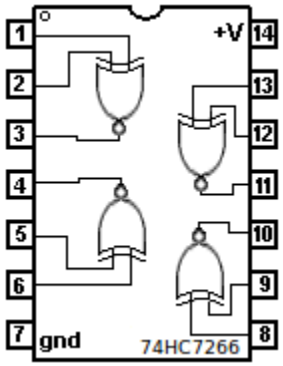
$$X = \overline{A \oplus B}$$

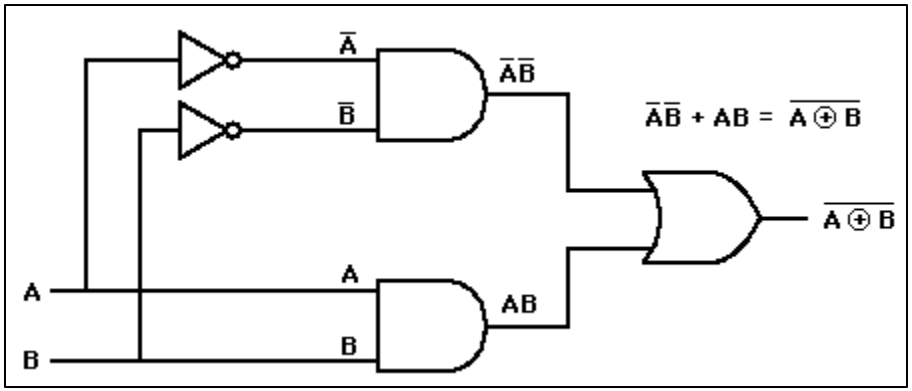
Logic Diagram Symbol



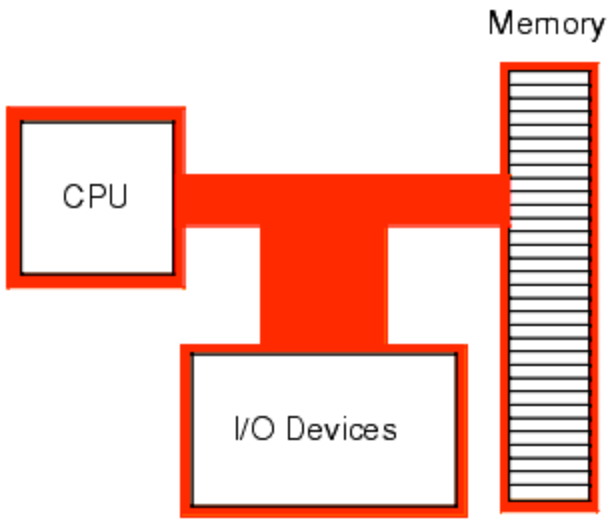
Truth Table

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

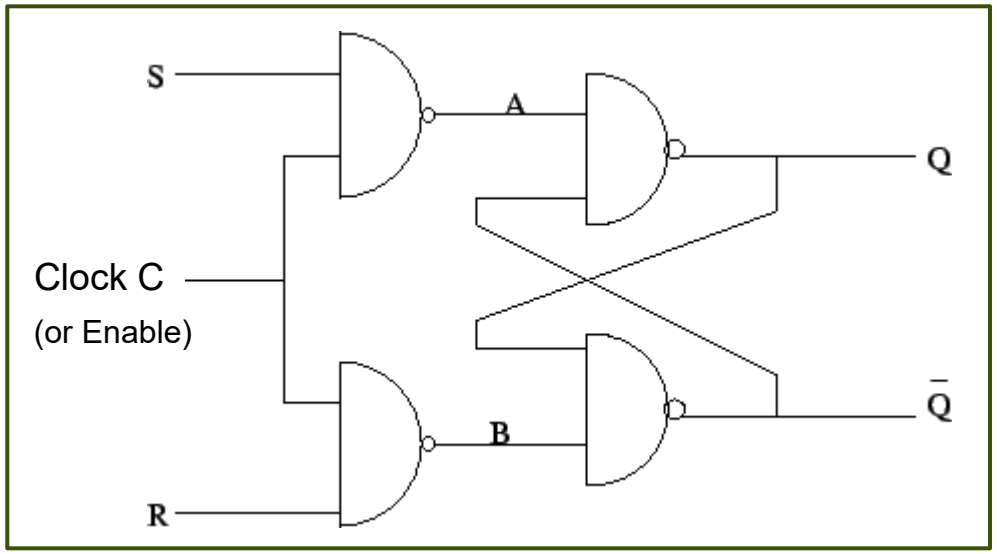


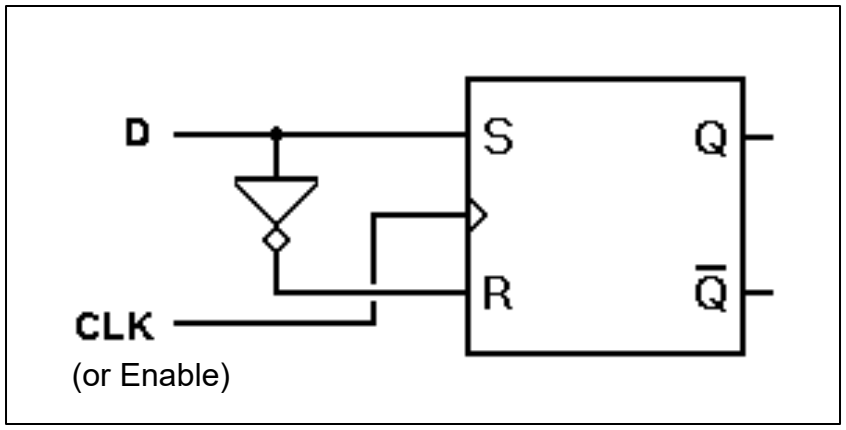


Register: Single Bit Device



D Flip-Flop

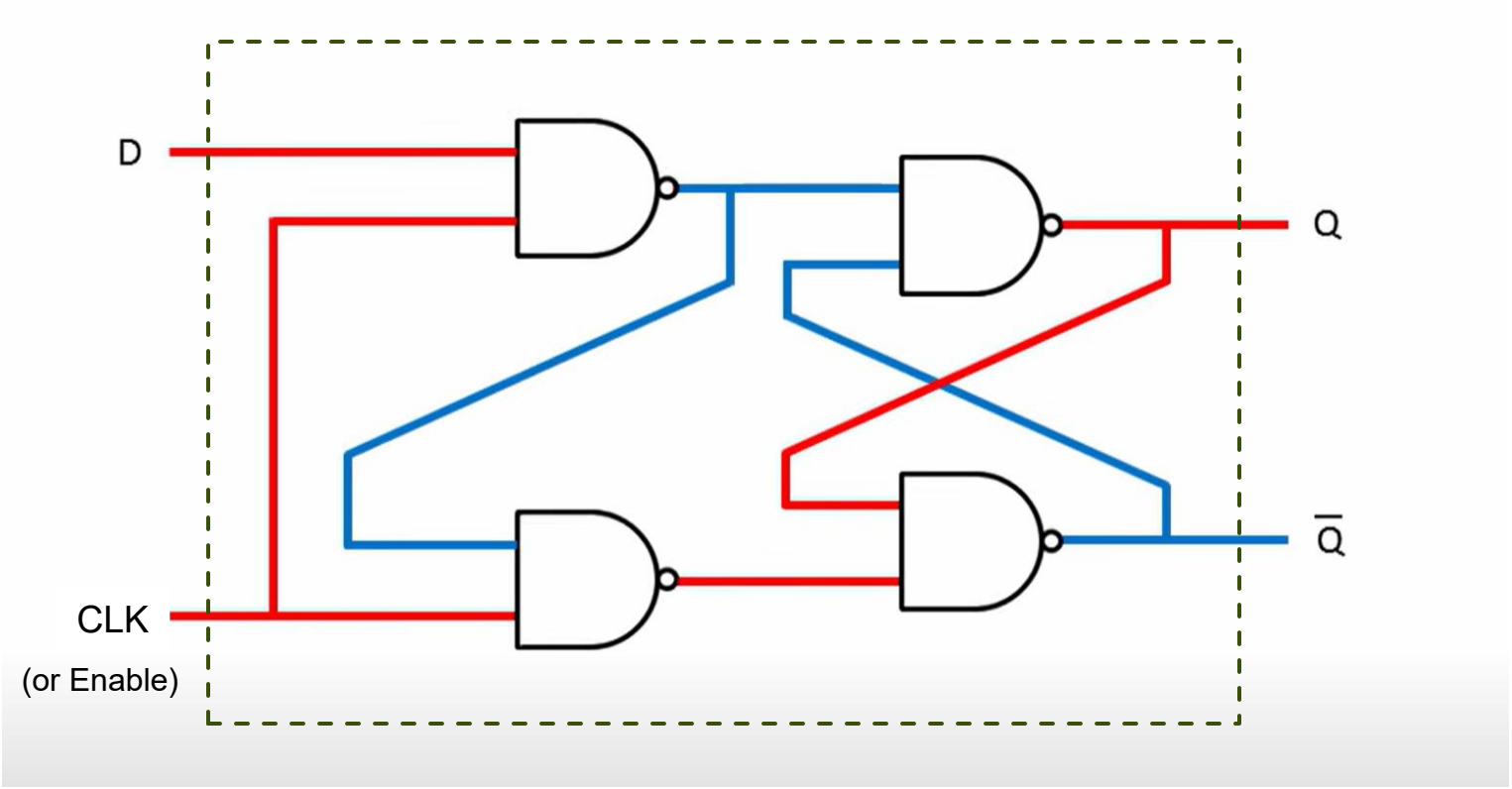




S	R	CLK	Q(t+1)	Comments
0	0	X	Q(t)	No change
0	1	↑	0	Reset
1	0	↑	1	Set
1	1	↑	?	Invalid

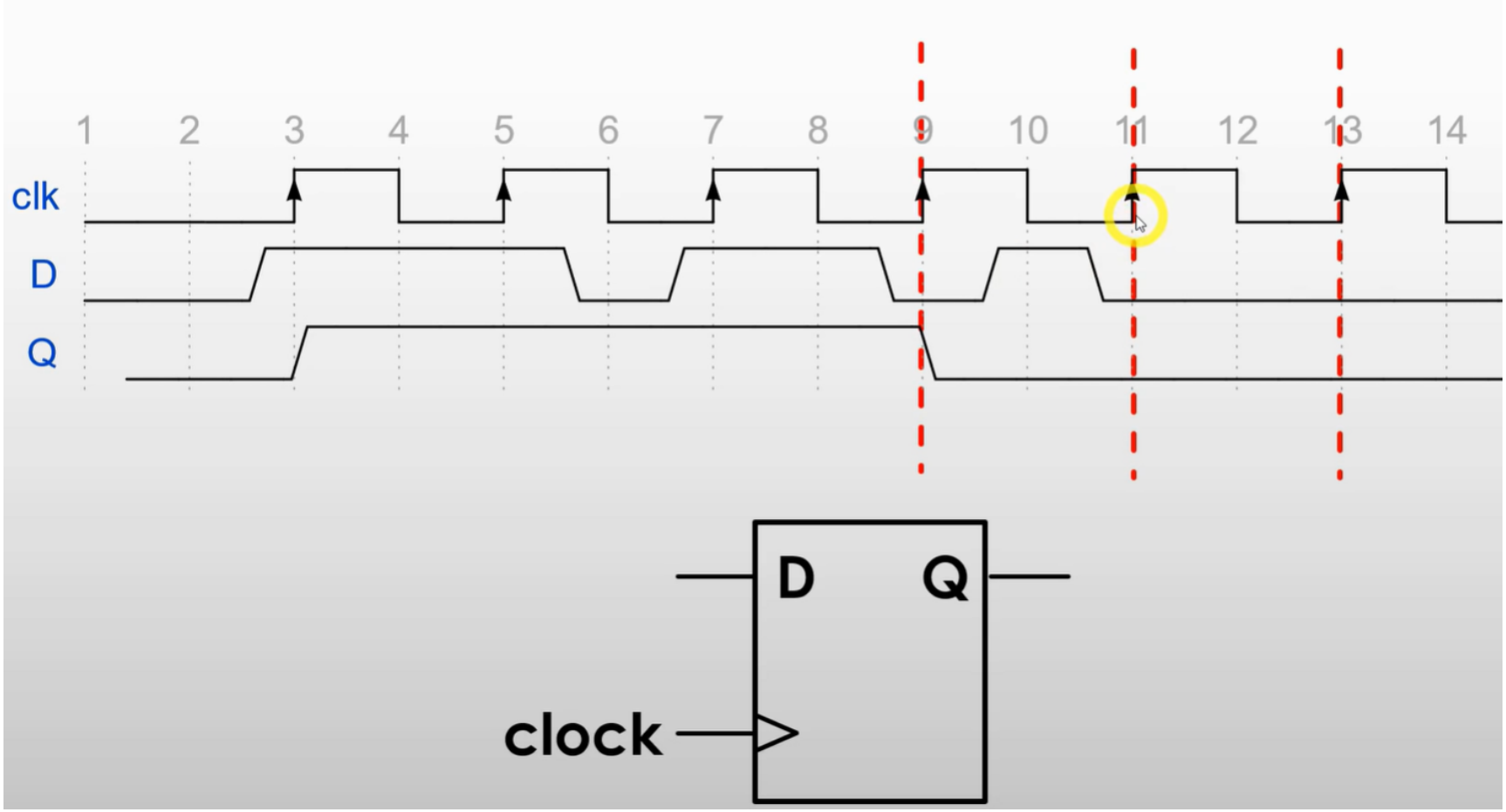
Rising Edge

Alternative Circuit of Gated D Latch …

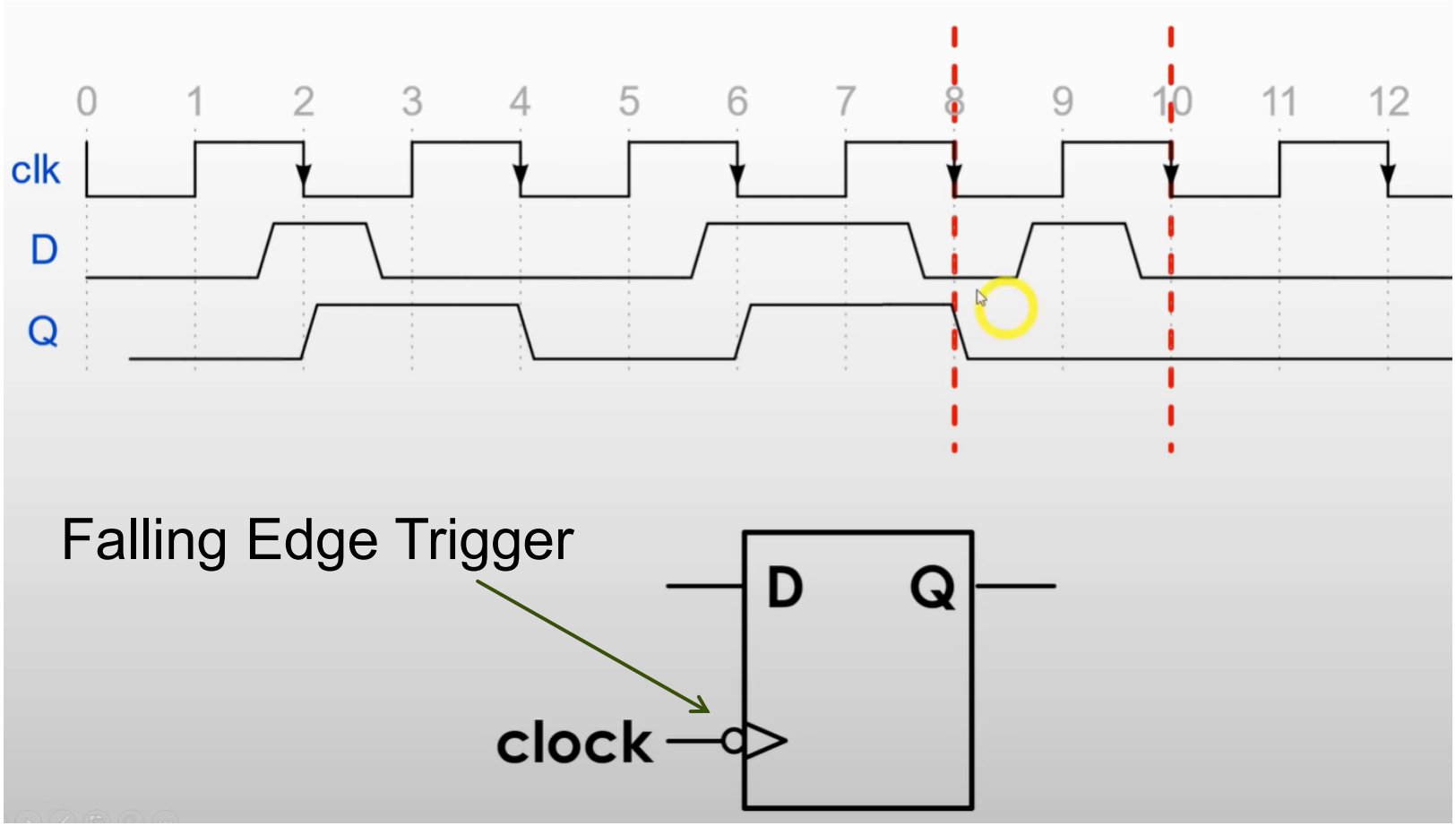


YouTube Video: https://youtu.be/y7Zf7Bv_J74

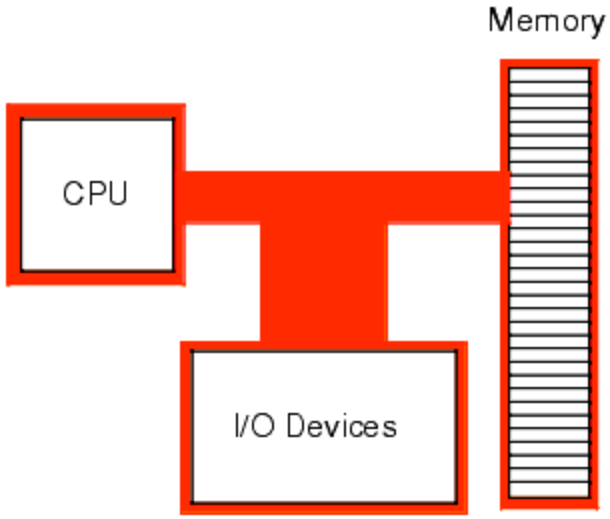
More Detail About Rising Edge Trigger …

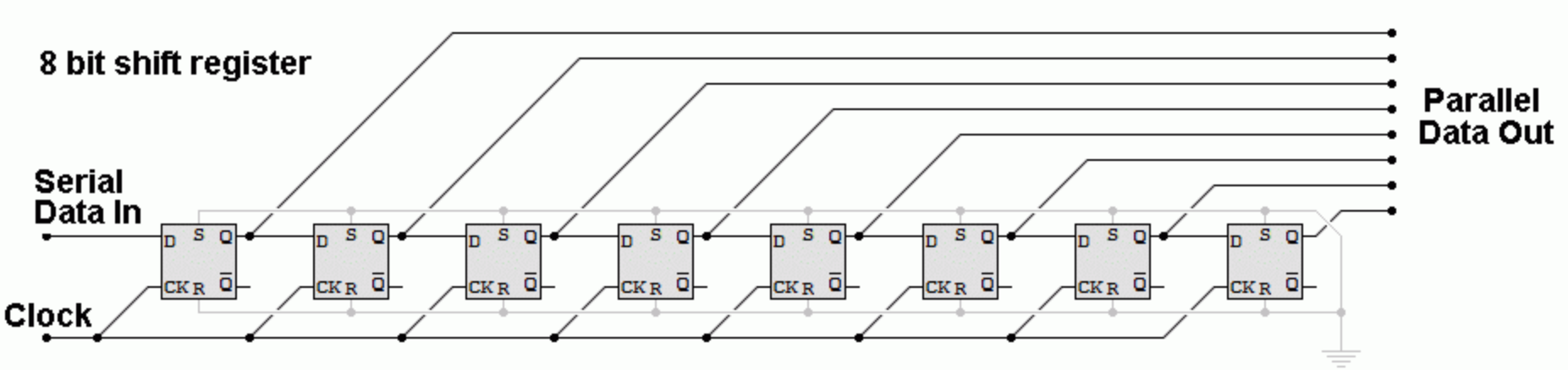


More Detail About Falling Edge Trigger …

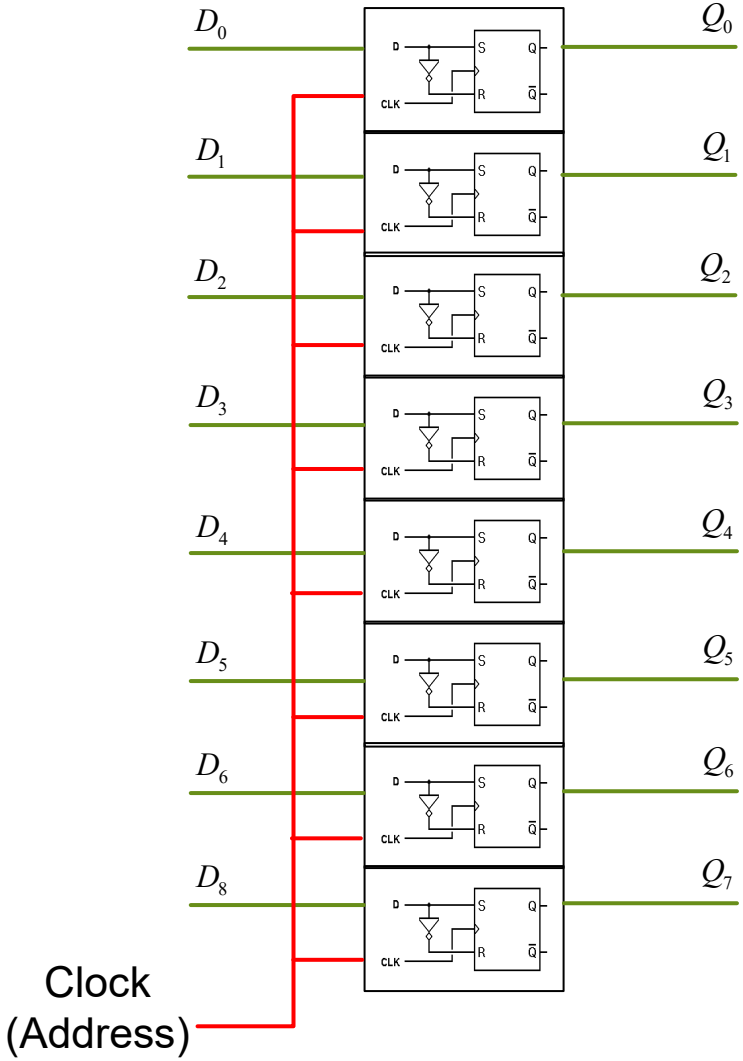


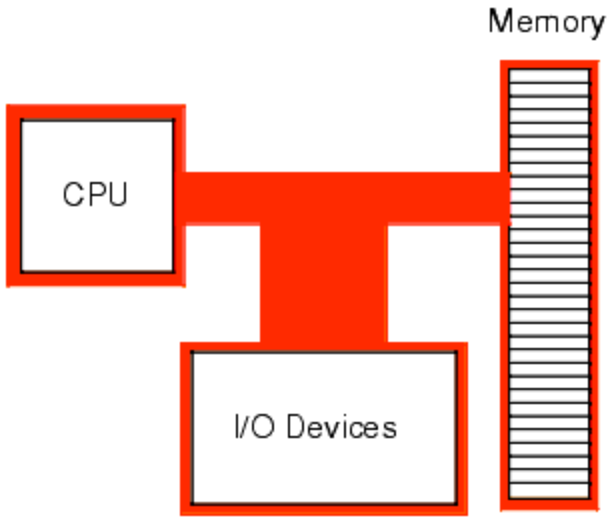
Register: 8-Bit Shift Register



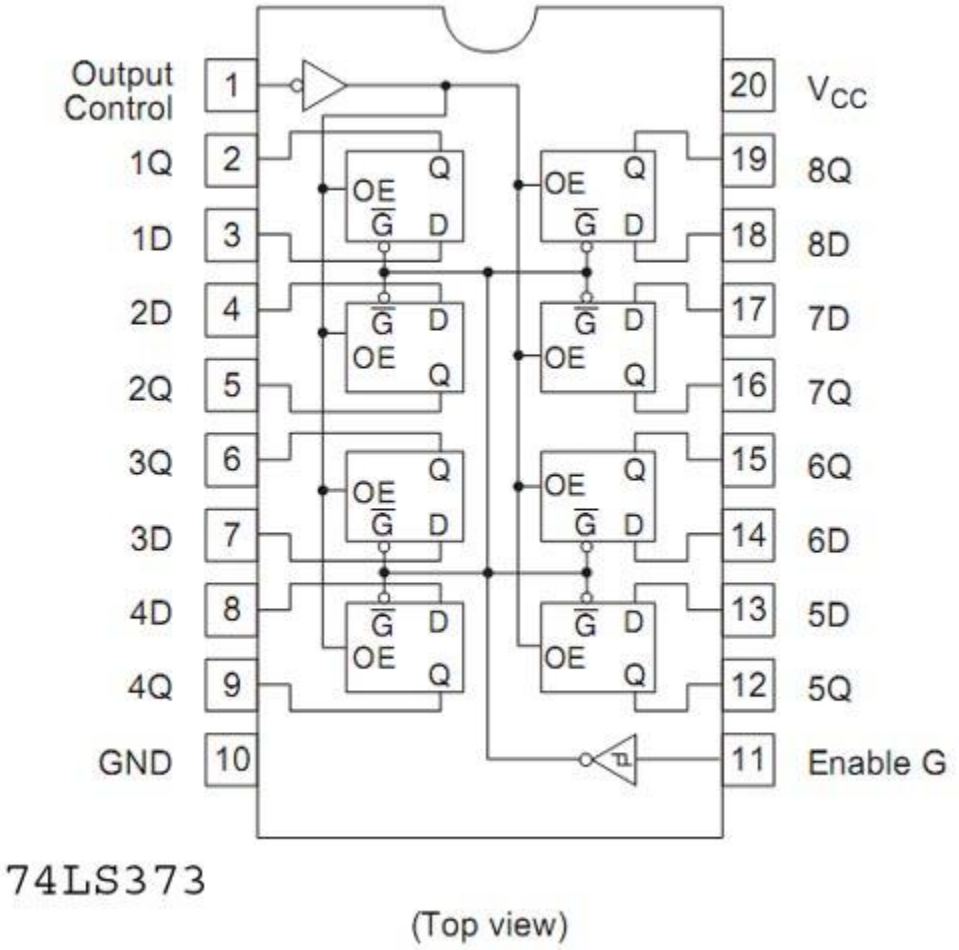


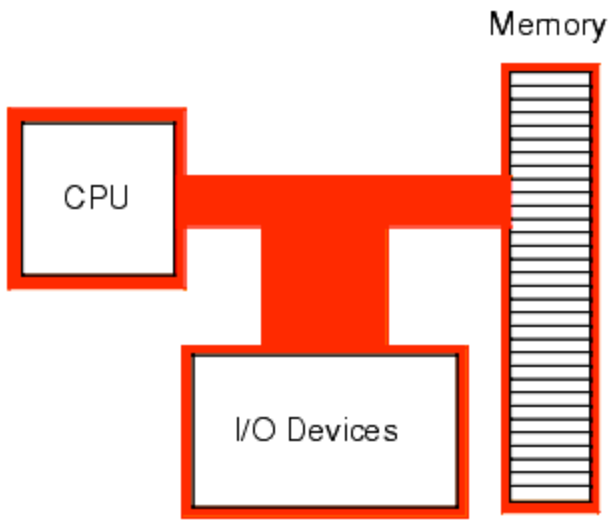
Register: 8-Bit Register (One Byte)



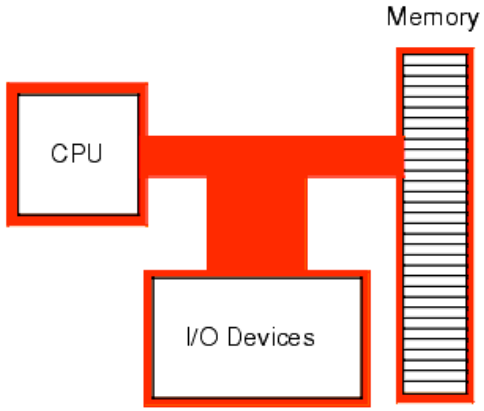


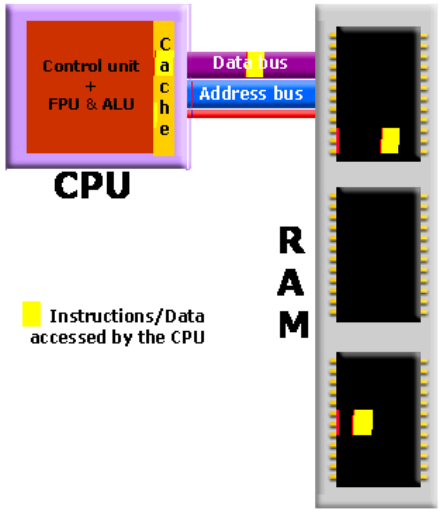
Example of 8-Bit Register (One Byte)



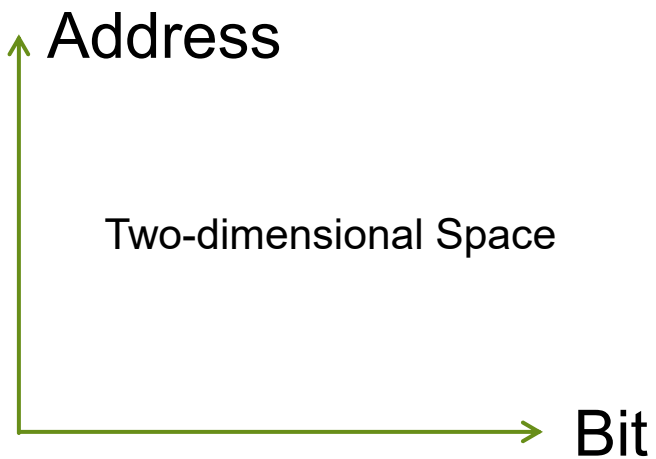


Outline





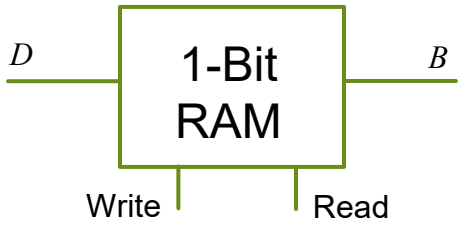
- Binary Logic Devices
- Memory Construction
- Memory Space in ARM
- Memory-Centric Operations
- Memory-Mapped I/O Devices

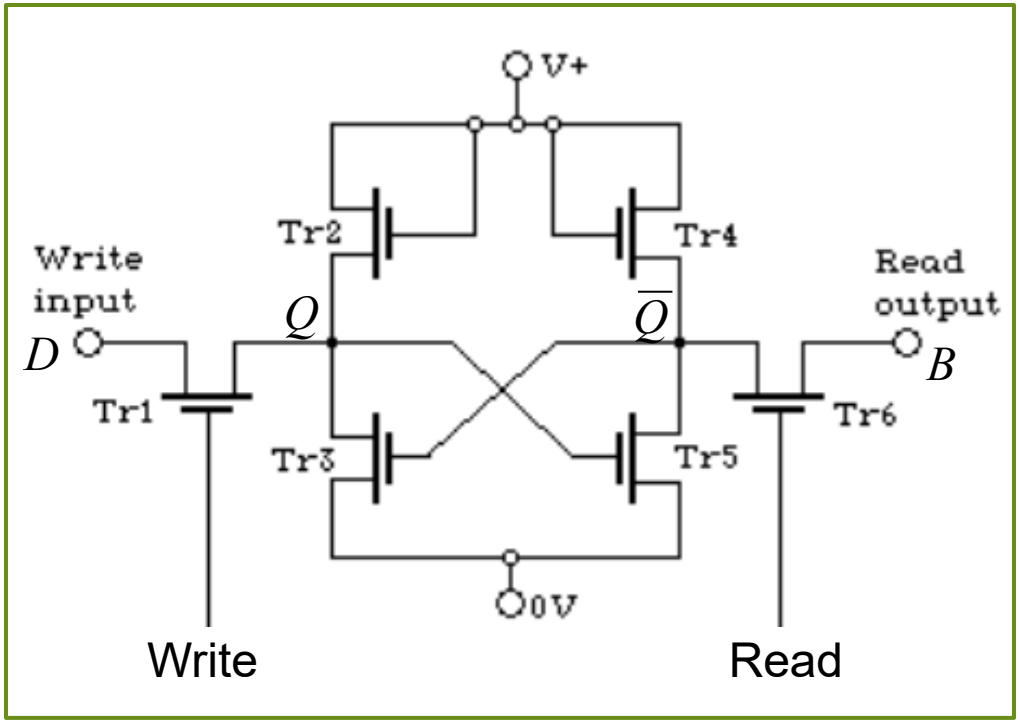


1. Memory Content
2. Memory Address
3. Memory Label

SRAM: Single Bit Static RAM

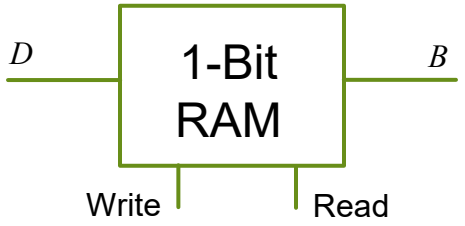
- Write Operation
  - "Write" line is in logic high.
  - Transistor Tr1 is on.
  - Q = D
- Read Operation
  - "Read" line is in logic high.
  - Transistor Tr5 is on.
  - B = 1 - Q



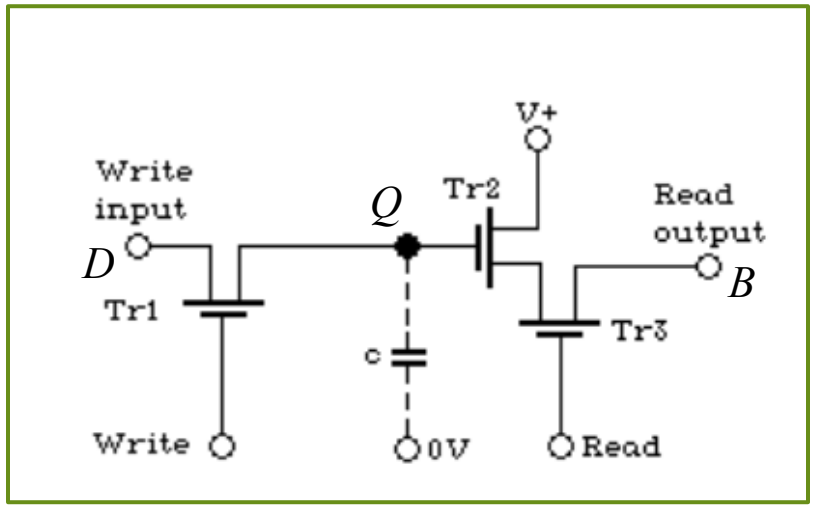


DRAM: Single Bit Dynamic RAM

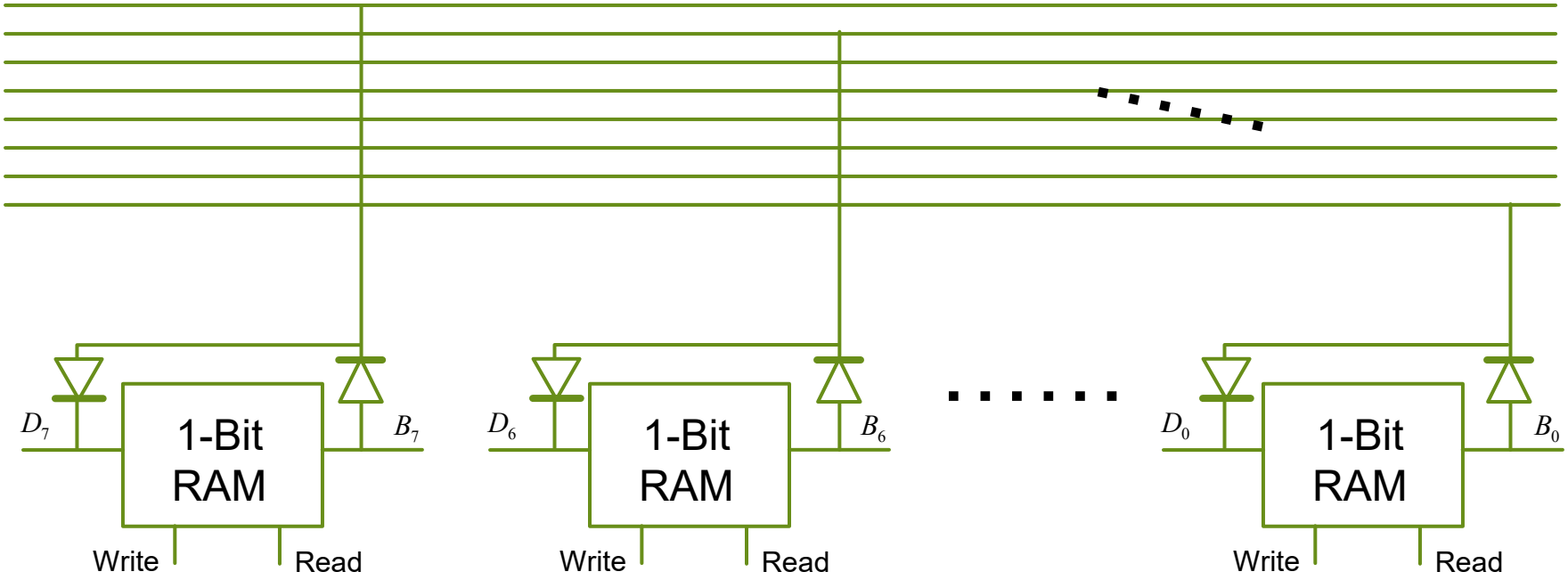
- Write Operation
  - "Write" line is in logic high.
  - Transistor Tr1 is on.
  - Q = D



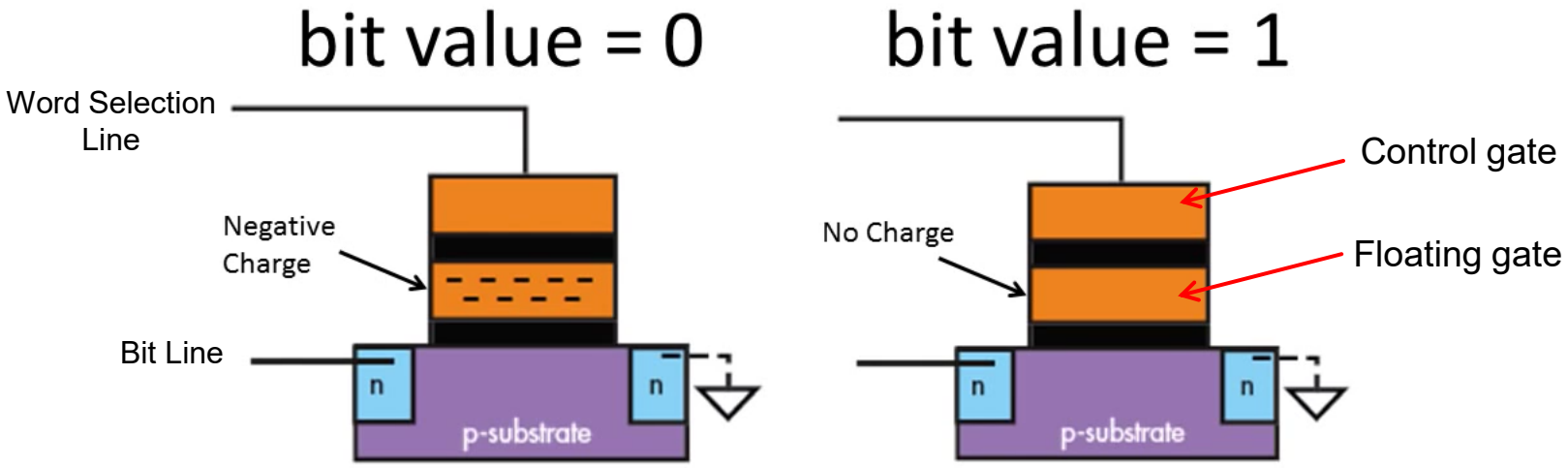
- Read Operation
  - "Read" line is in logic high.
  - Transistor Tr3 is on.
  - B = Q



RAM: 8-Bit RAM (One Byte)



Flash Memory: 1-Bit Flash Cell

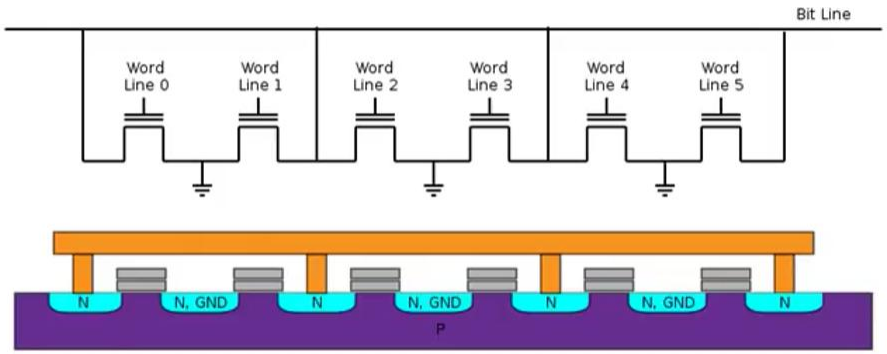


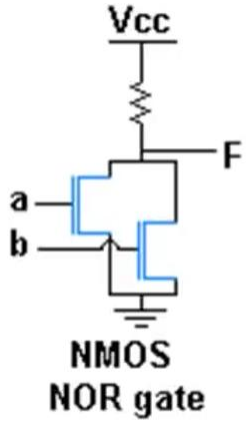
SLC (Single Level Cell)

- 2 charge states
- 1 bit per floating gate transistor

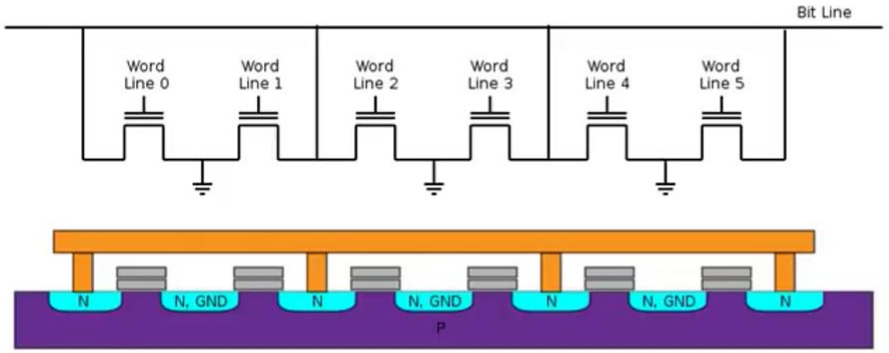
Example

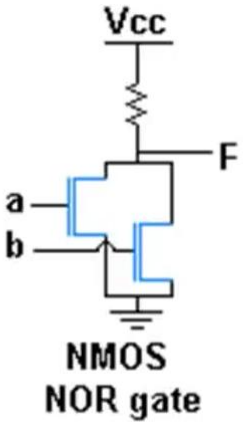
Bit 1



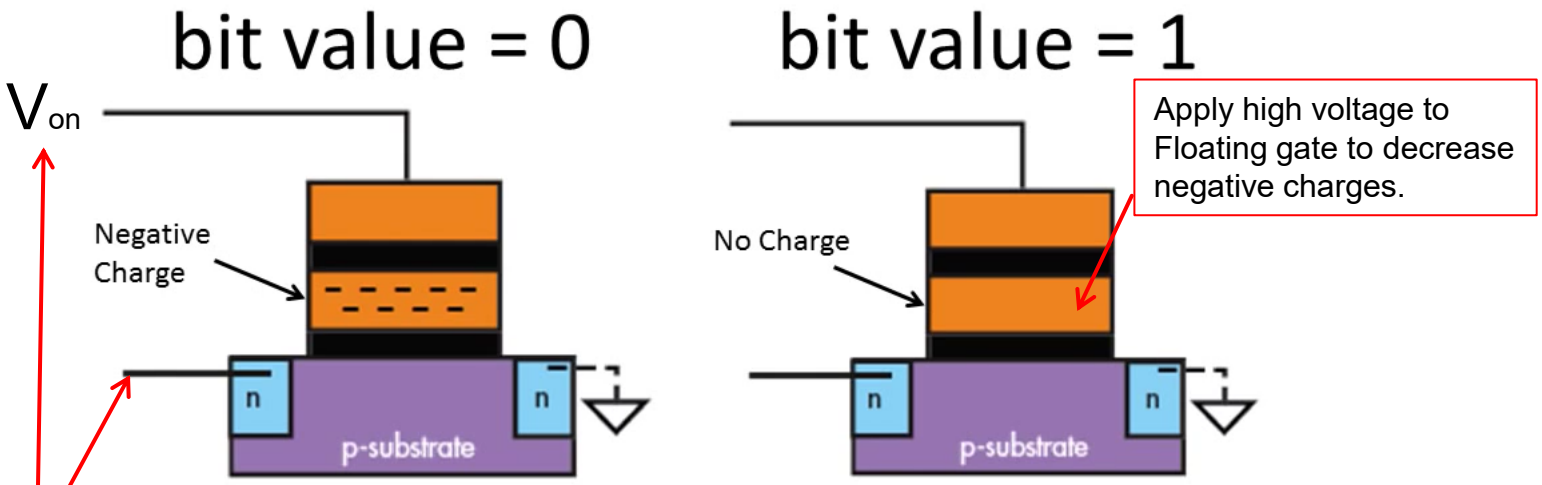


Bit 0





Flash Memory: Write of 1-Bit Flash Cell



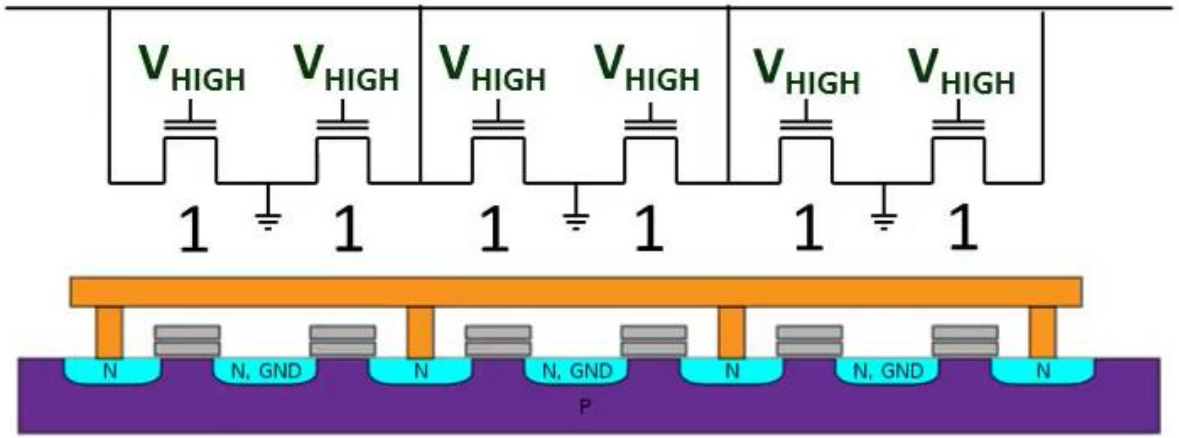
SLC (Single Level Cell)

- 2 charge states
- 1 bit per floating gate transistor

Example

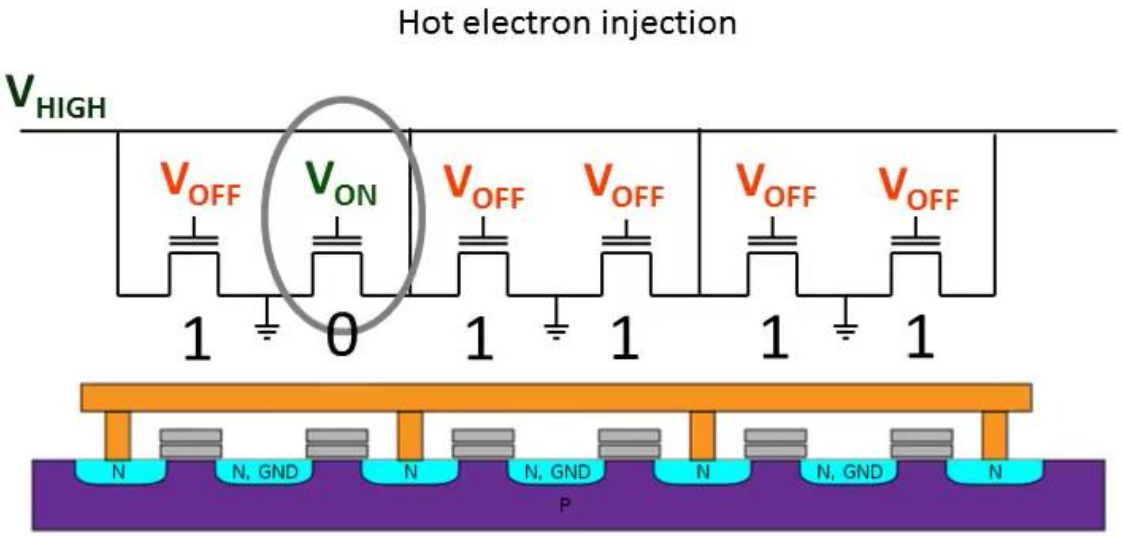
NOR Erasing (Set all bits to 1)

Block erasure via tunneling



Example

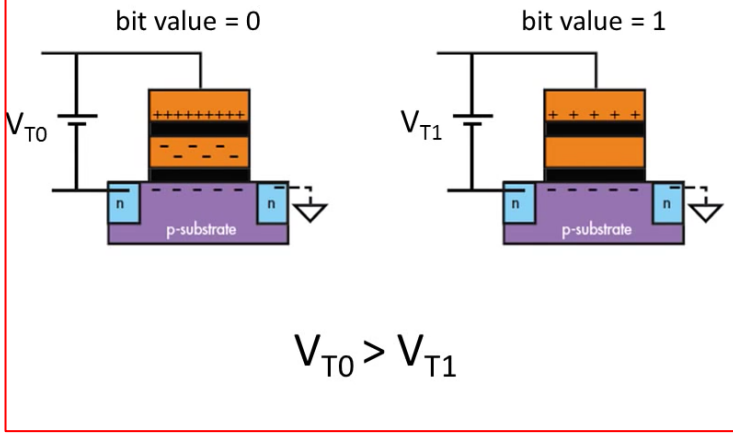
NOR Writing (Setting 0s)

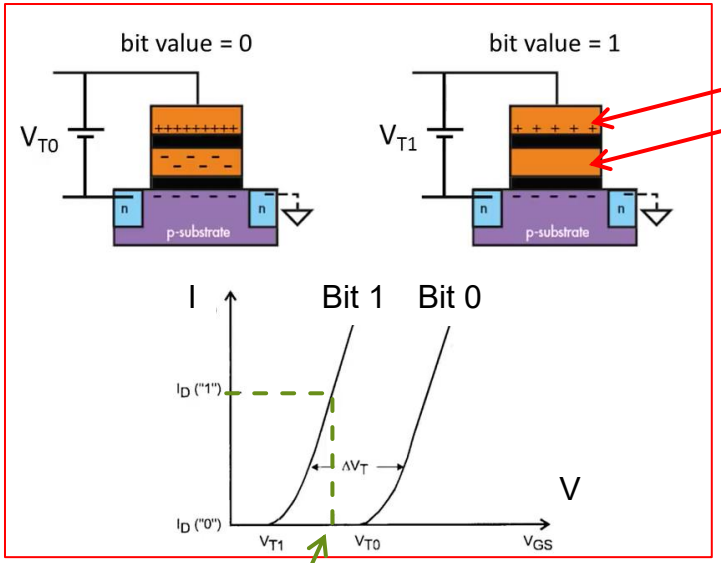


Flash Memory: Read of 1-Bit Flash Cell

Reading Results:
There is no current -> Output is Value 0
There is Current -> Output is Value 1

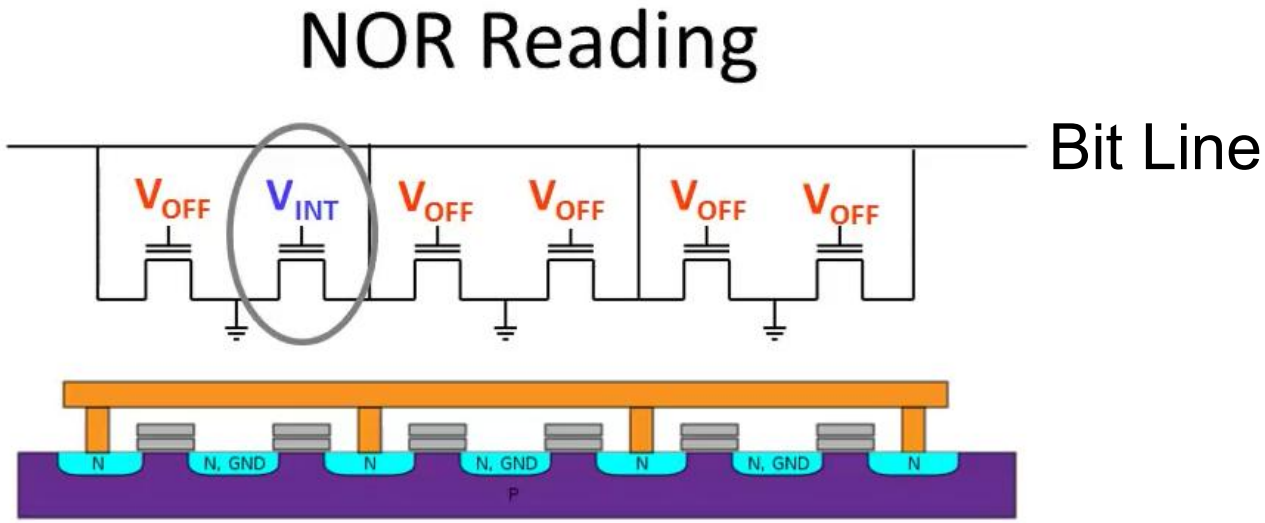
Value 0 = high voltage
Value 1 = low voltage

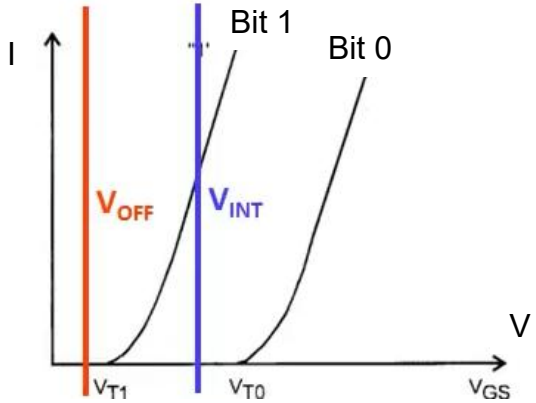




$$V_{T1} > V_{GS} > V_{T0}$$

Example: Reading one bit





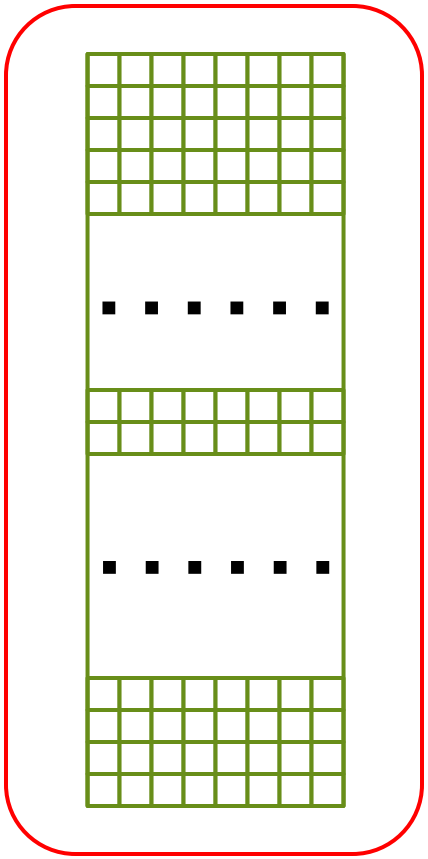
Reading Results:
There is no current -> Output is Value 0
There is Current -> Output is Value 1

$$V_{T1} > V_{GS} > V_{T0}$$

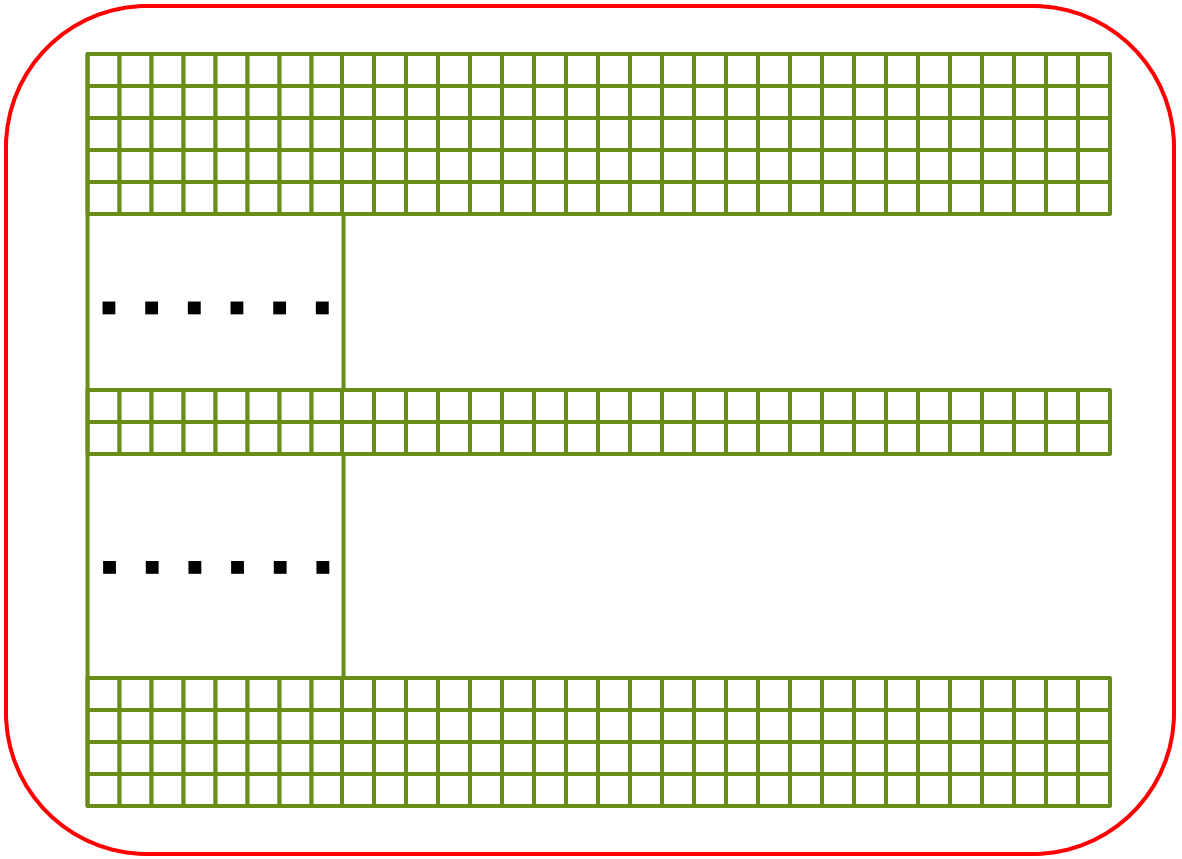
Blueprint of Memory Unit

- A memory unit consists of a set of bytes. Each byte has eight bits. Each bit has one memory cell for read and write operations.

Memory with 8-bit Data Bus

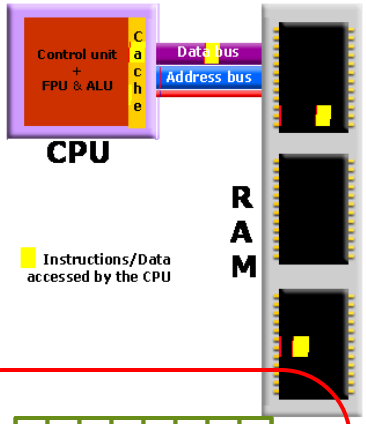


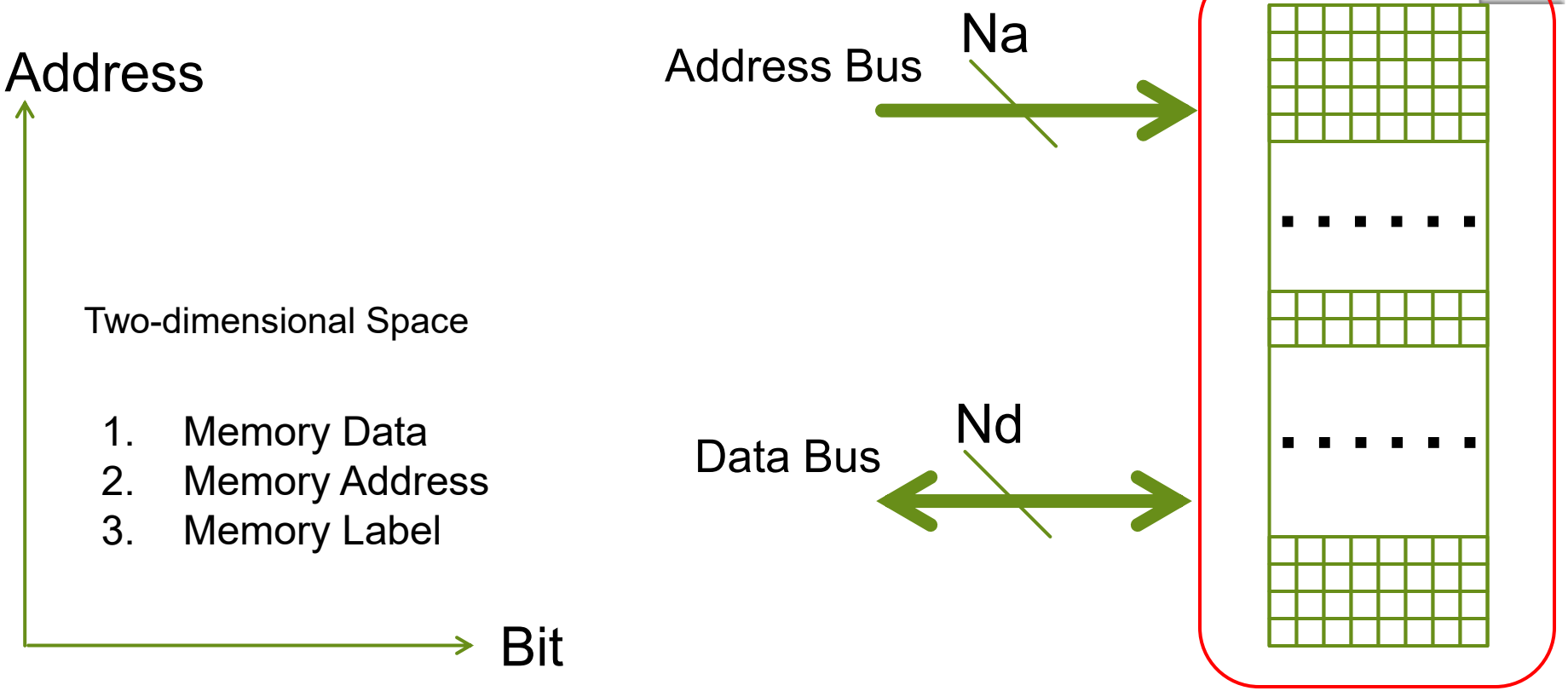
Memory with 32-bit Data Bus



Input to Memory Unit

- Input of memory unit includes:
  - a) address of a byte, and
  - b) data of one or more bytes.

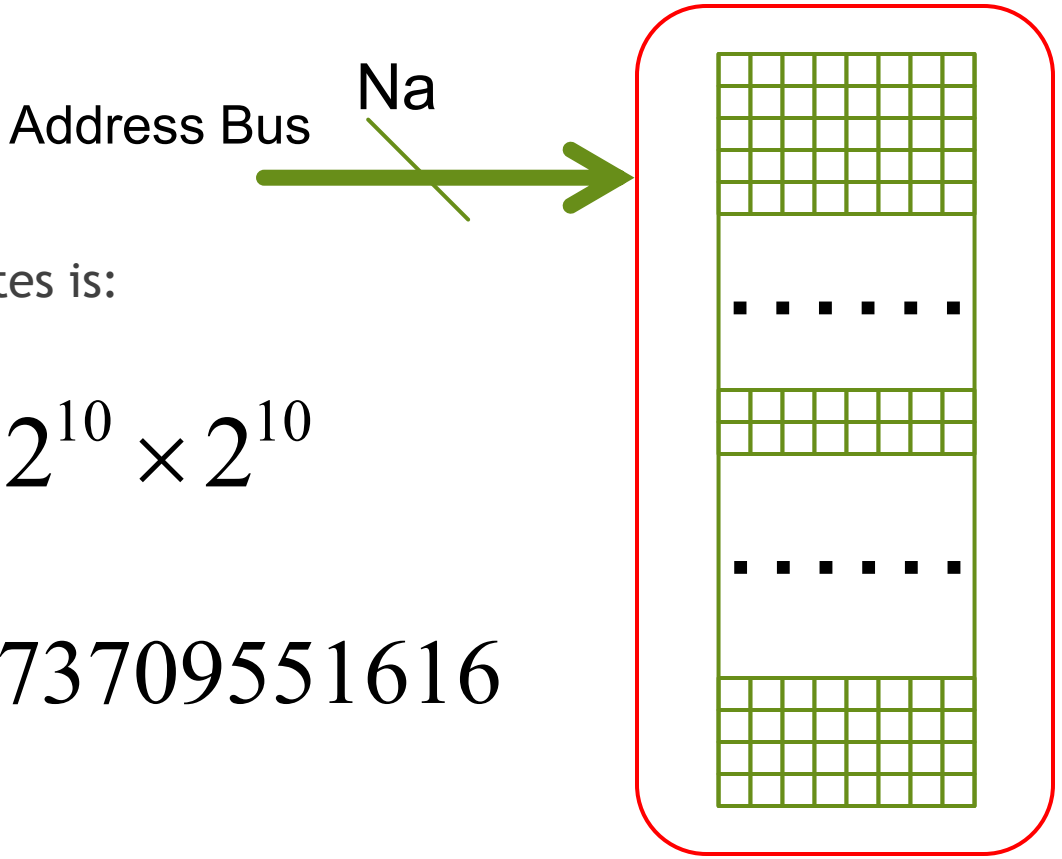




Exercise

- A memory unit's address bus has 64 bits. Hence, Na = 64. What is the maximum number of bytes that the memory unit could have?

- Answer:



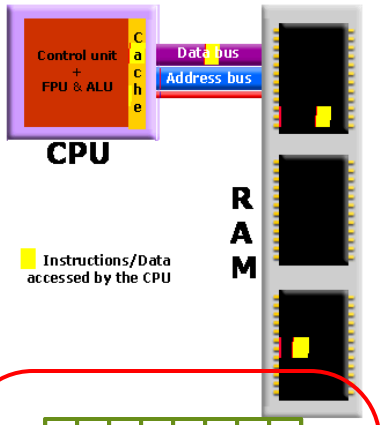
- The maximum number of bytes is:

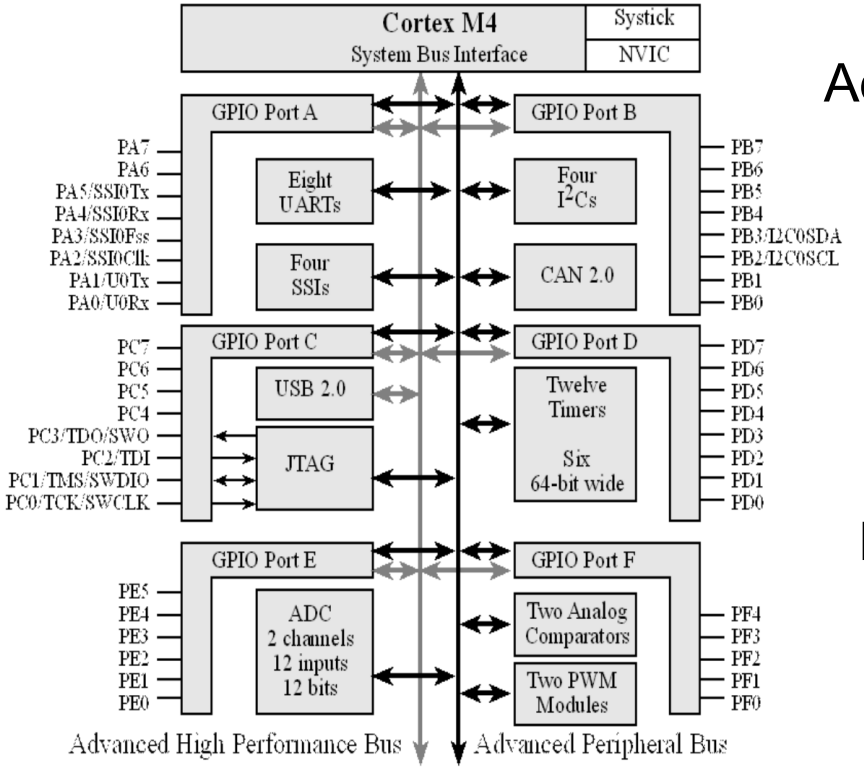
$$2^{64} = 2^{34} \times 2^{10} \times 2^{10} \times 2^{10}$$

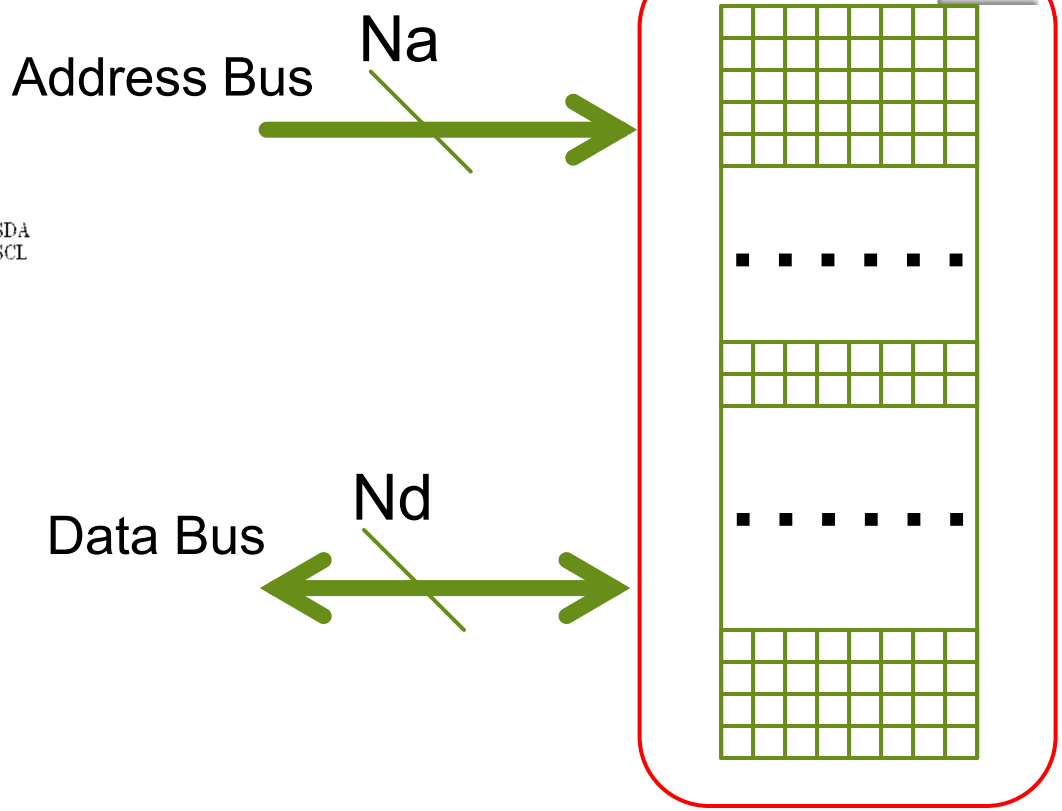
$$2^{64} = 18446744073709551616$$

Output from Memory Unit

- The output from memory unit is data of one or more bytes.



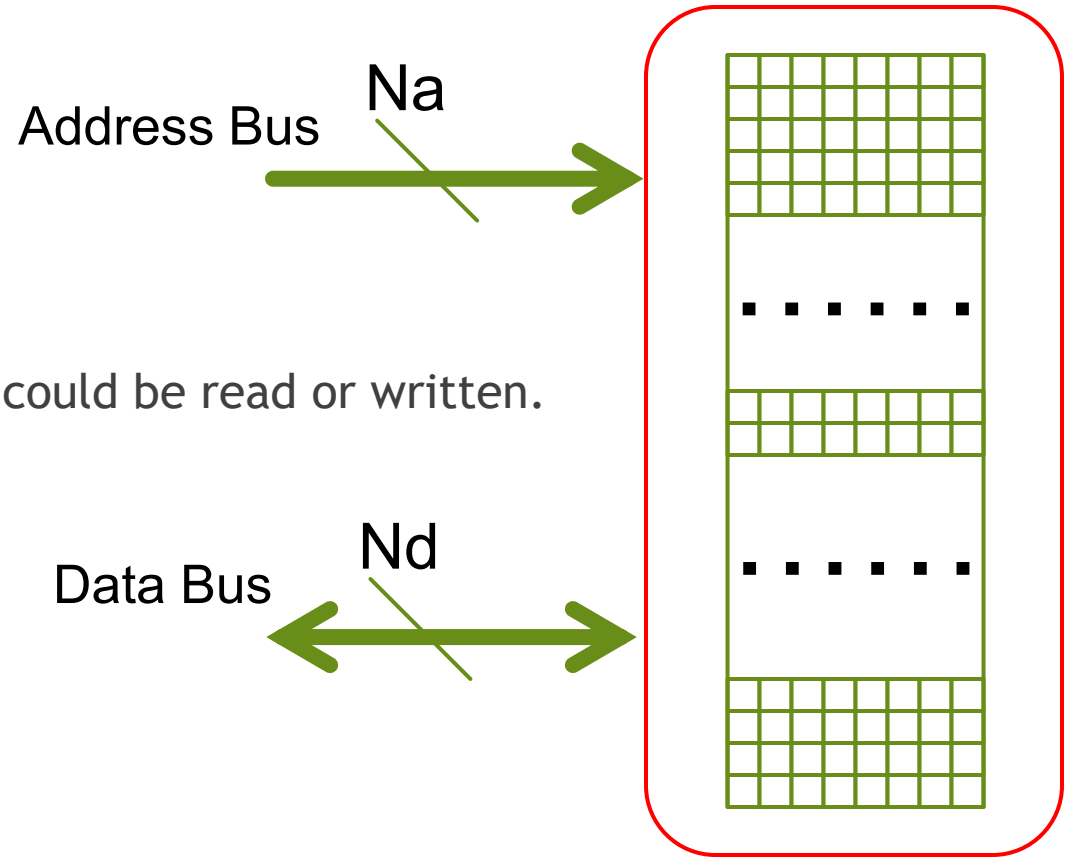




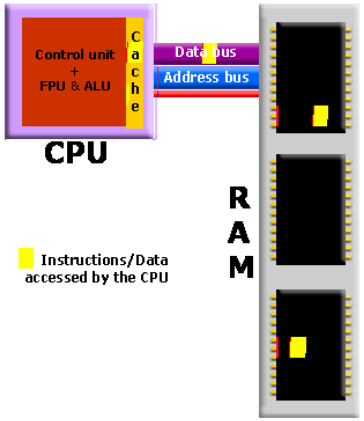
Exercise

- A memory unit's address bus has 64 lines. And, its data bus has 32 lines. Hence, Na = 64 and Nd = 32. At one time, how many bytes could the memory unit read or write?

- Answer:



- At one time, 32/8 = 4 bytes could be read or written.

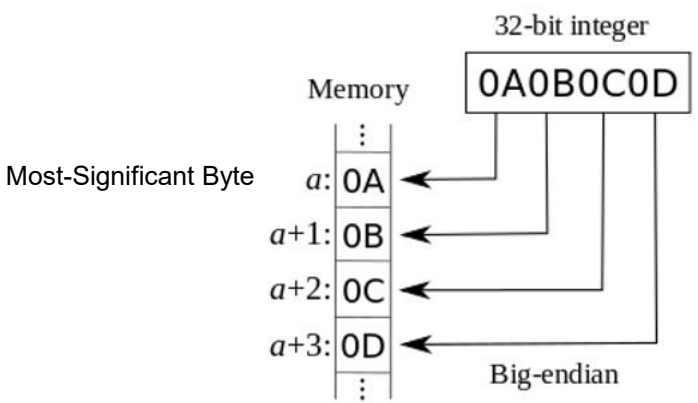


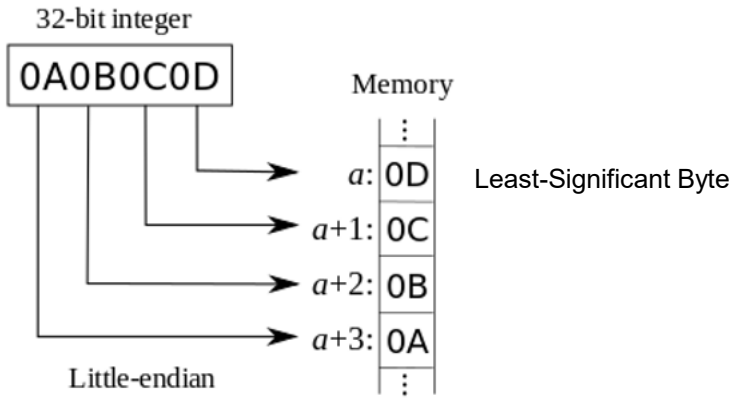
Endianness of Memory

- **Endianness** is the ordering or sequencing of bytes of a word of digital data in computer memory storage or during transmission. Words may be represented in **big-endian** or **little-endian** manner. Big-endian systems store the most-significant byte of a word at the smallest memory address and the least significant byte at the largest. A little-endian system, in contrast, stores the least-significant byte at the smallest address.

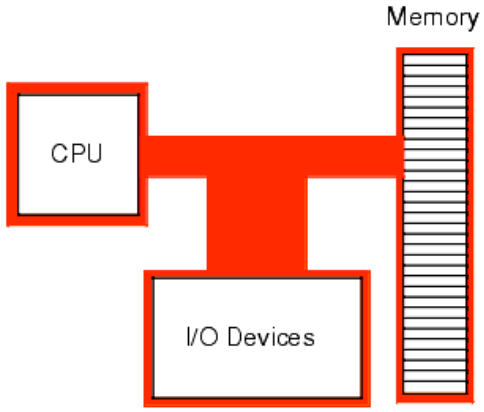
- Example: (ARM with Linux operating system follows Little endian)

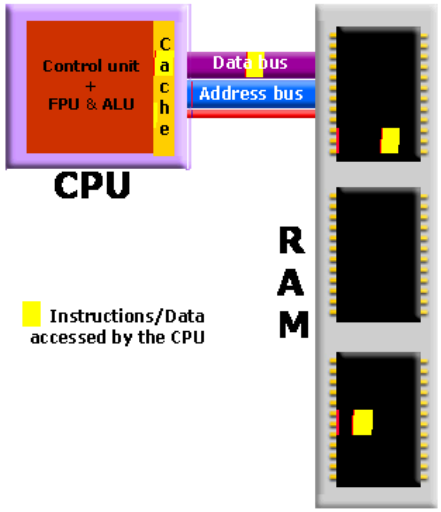
(Imagine: A super room has four small rooms. How to determine the small rooms' keys?)



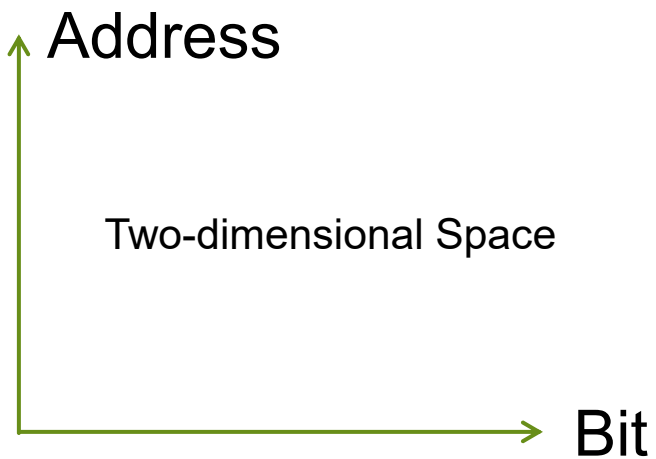


Outline



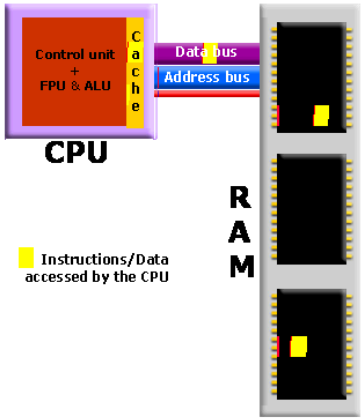


- Binary Logic Devices
- Memory Construction
- Memory Space in ARM
- Memory-Centric Operations
- Memory-Mapped I/O Devices

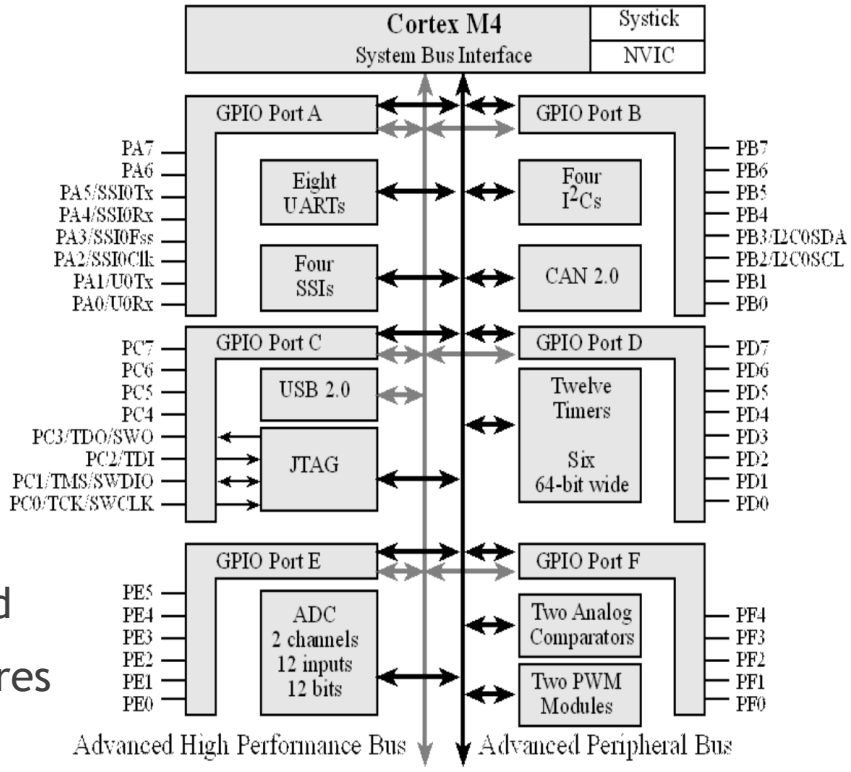


1. Memory Content
2. Memory Address
3. Memory Label

Terminology of ARM

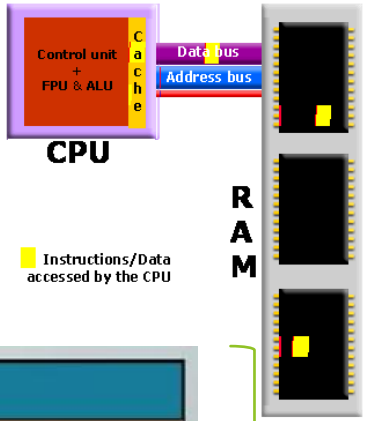


- ARM is a RISC architecture
  - ISA stands for Instruction Set Architecture which are innate
  - RISC stands for Reduced Instruction Set Computer
  - Most instructions execute in a single cycle
  - ARM Instruction Set is 32-bit
  - Thumb Instruction Set is 16/32-bit
- ARM is a 32-bit load-store architecture
  - 8 bits are called a Byte
  - 16 bits or two bytes are called a Half Word
  - 32 bits or four bytes are called a Word
  - 64 bits or eight bytes are called Double Word
  - The only memory accesses are loads and stores



Operation Modes of ARM

- ARM Core has seven basic modes



Mode	Description
Supervisor (SVC)	Entered on reset and when a Supervisor call instruction (SVC) is executed
FIQ	Entered when a high priority (fast) interrupt is raised
IRQ	Entered when a normal priority interrupt is raised
Abort	Used to handle memory access violations
Undef	Used to handle undefined instructions
System	Privileged mode using the same registers as User mode
User	Mode under which most Applications / OS tasks run

Exception Modes

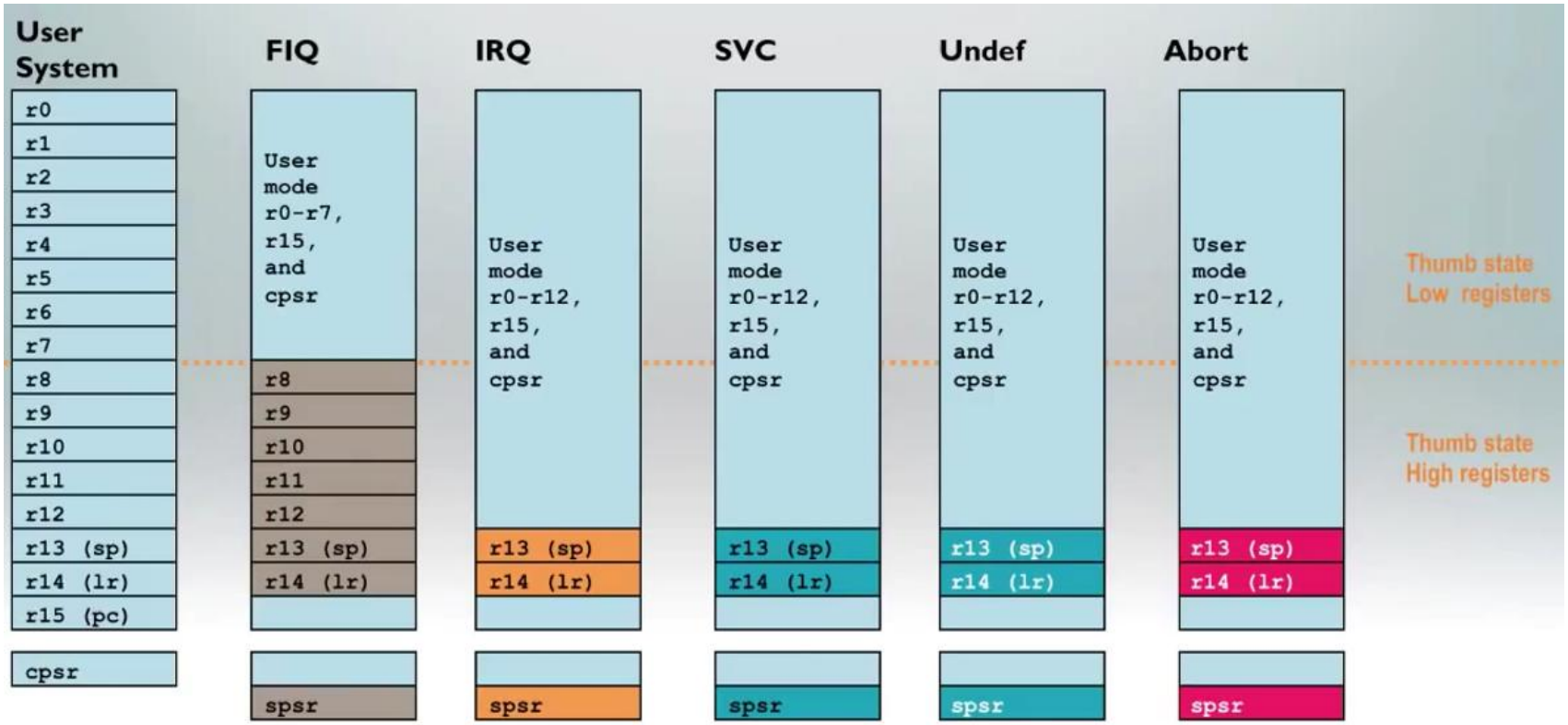
Privileged Modes

Unprivileged Mode

Registers of ARM (1)

- ARM Core has a large number of registers

PC: Program Counter
LP: Link Register (saved value of PC)
SP: Stack Pointer
CPSR: Current Program Status Register
SPSR: Saved Program Status Register



Note: System mode uses the User mode register set

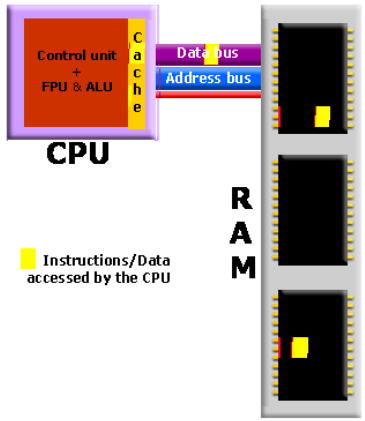
Registers of ARM (2)

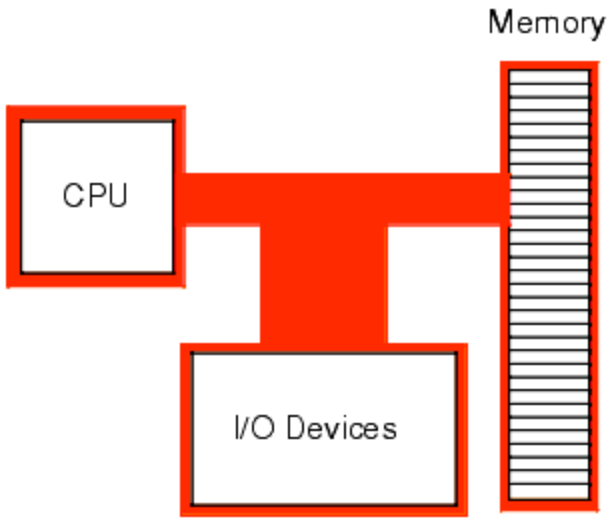
- One program counter register (PC), 30 general purpose registers and 6 status registers

User/System	Supervisor	Abort	Undefined	Interrupt	Fast interrupt
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8_FIQ
R9	R9	R9	R9	R9	R9_FIQ
R10	R10	R10	R10	R10	R10_FIQ
R11	R11	R11	R11	R11	R11_FIQ
R12	R12	R12	R12	R12	R12_FIQ
R13	R13_SVC	R13_ABORT	R13_UNDEF	R13_IRQ	R13_FIQ
R14	R14_SVC	R14_ABORT	R14_UNDEF	R14_IRQ	R14_FIQ
PC	PC	PC	PC	PC	PC

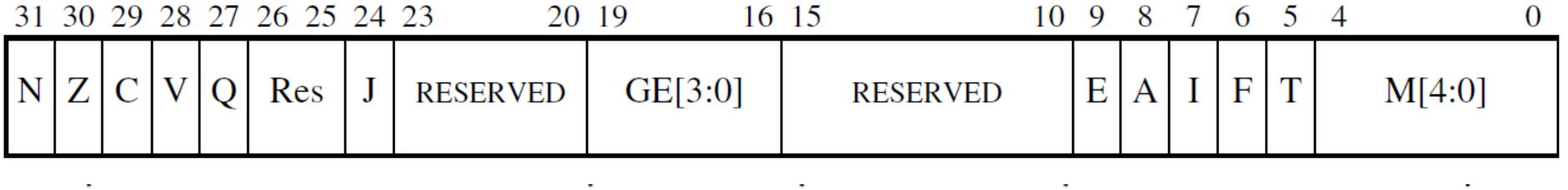
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_SVC	SPSR_ABORT	SPSR_UNDEF	SPSR_IRQ	SPSR_FIQ

= *banked register*





Program Status Register: R16



Flag Field	
N	Negative result from ALU
Z	Zero result from ALU
C	ALU operation caused Carry
V	ALU operation oVerflowed
Q	ALU operation saturated
J	Java Byte Code Execution

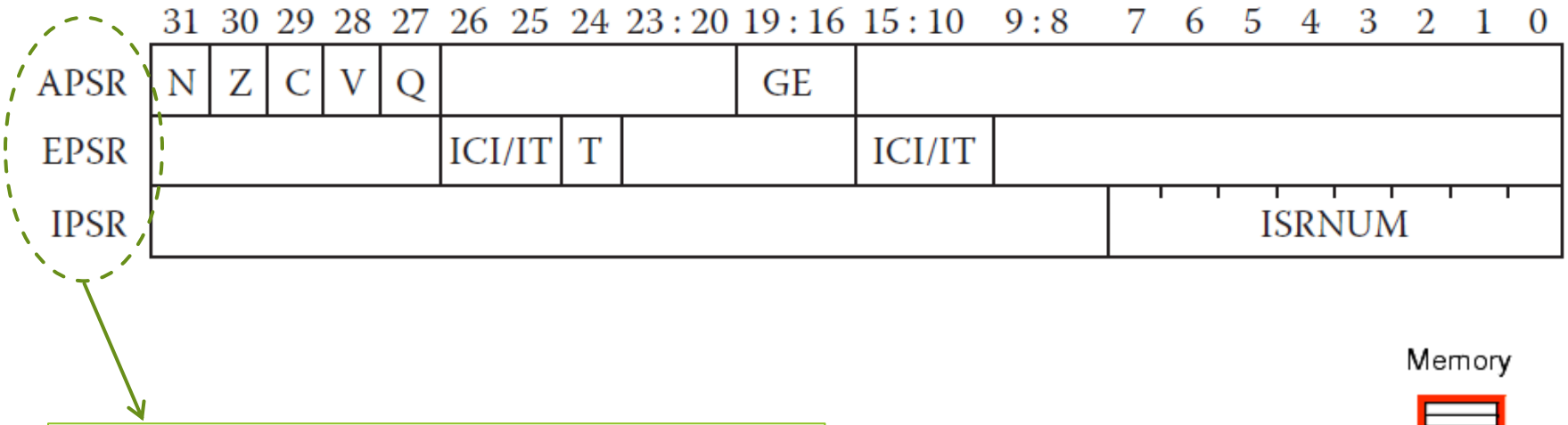
GE: Greater than or Equal Status Flags

Control bits	
I	1: disables IRQ
F	1: disables FIQ
T	1: Thumb, 0: ARM

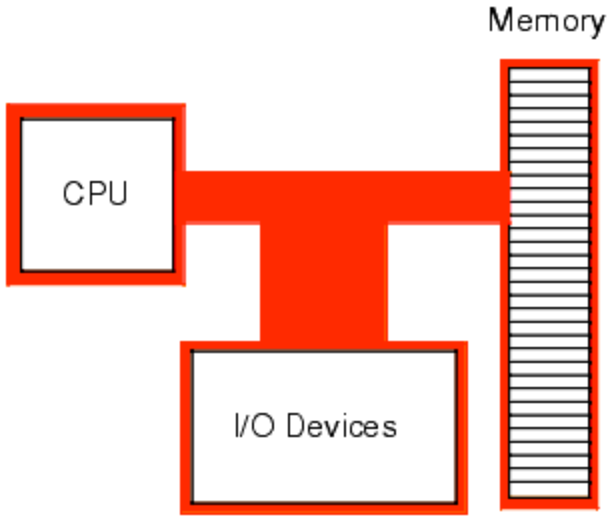
Mode bits M[4:0}	
0b10000	User
0b11111	System
0b10001	FIQ
0b10010	IRQ
0b10011	SVC(Supervisor)
0b10111	Abort
0b11011	Undefined

Cortex M4's Status Register

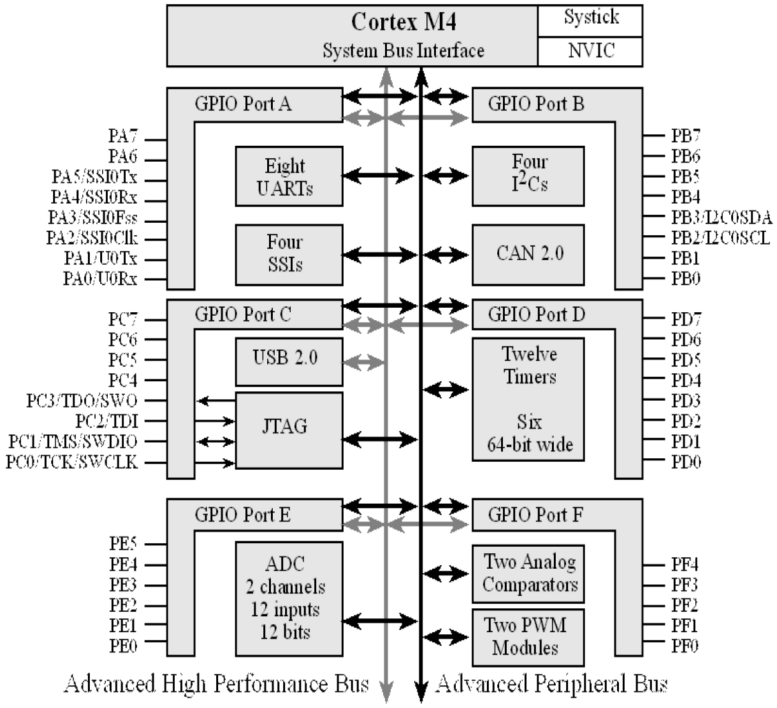
GE: Greater than or Equal Status Flags

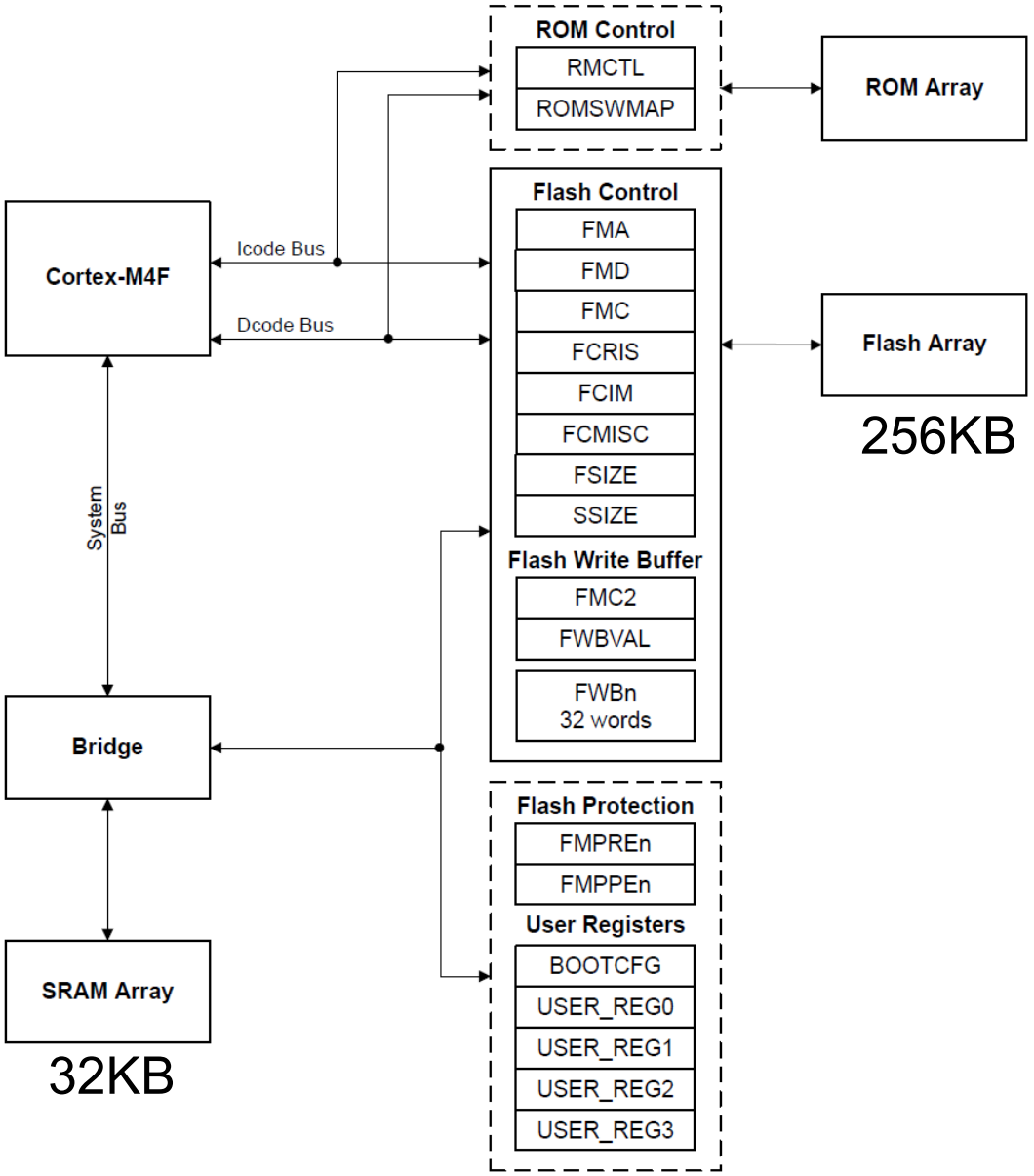


APSR: Application Program Status Register
EPSR: Execution Program Status Register
IPSR: Interrupt Program Status Register

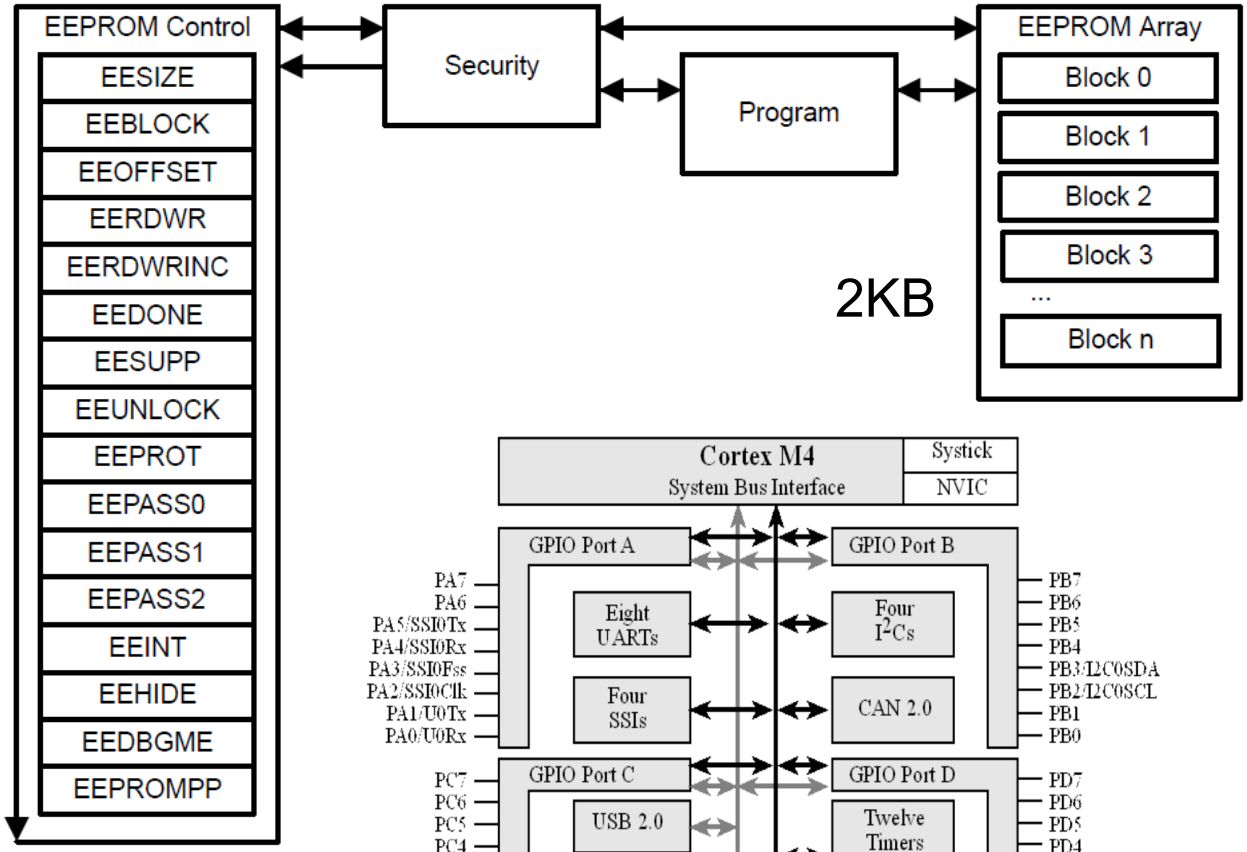


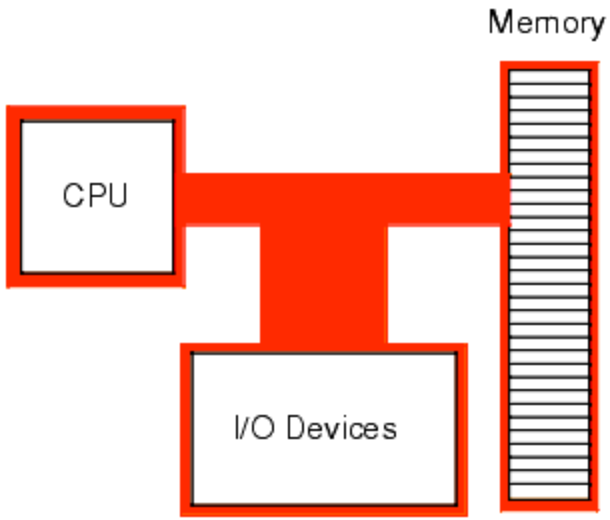
Memory Inside TM4C123G





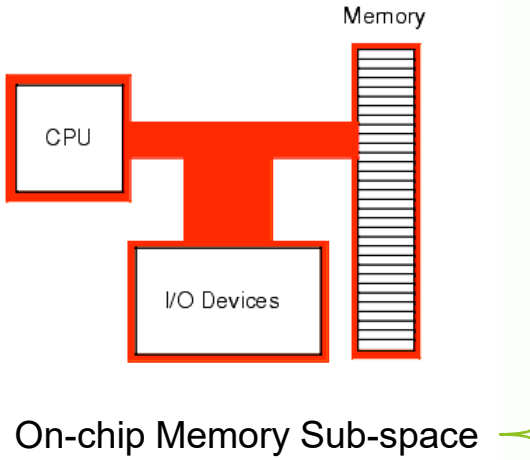
Memory Inside TM4C123G

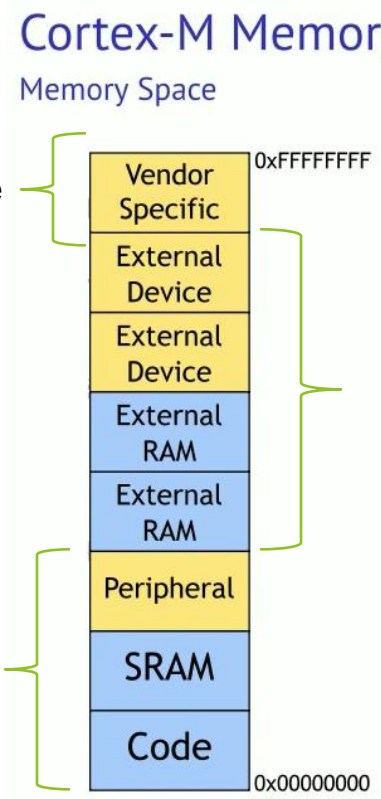


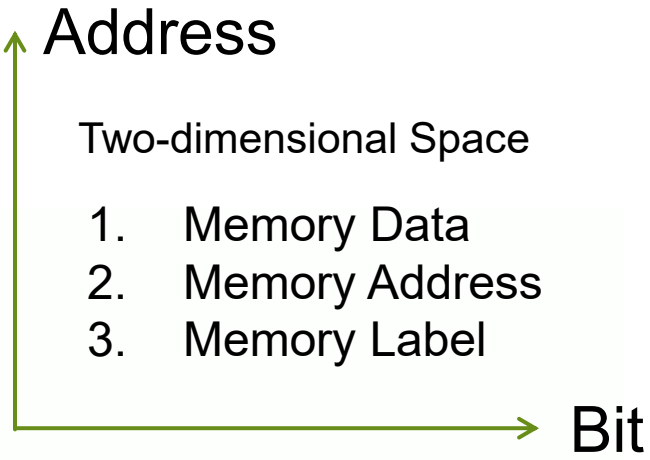


ARM Cortex-M's Memory Space

- On-chip Memory Sub-space
- Off-chip Memory Sub-space
- Private Memory Sub-space



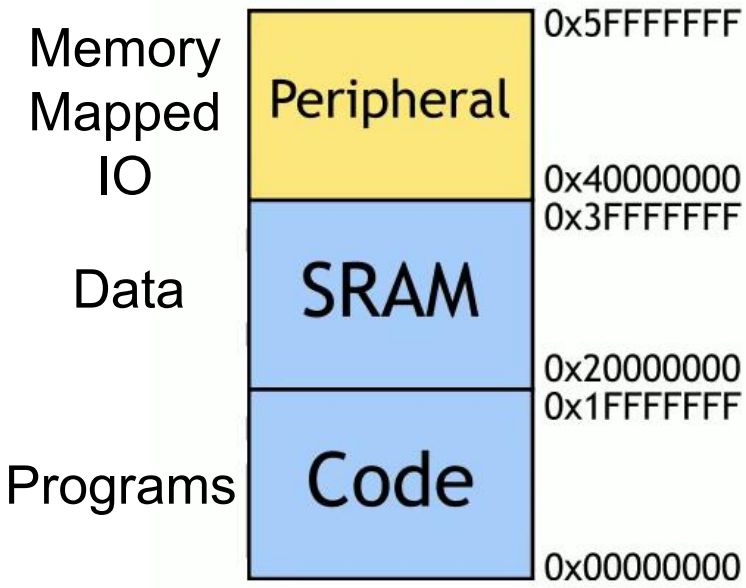


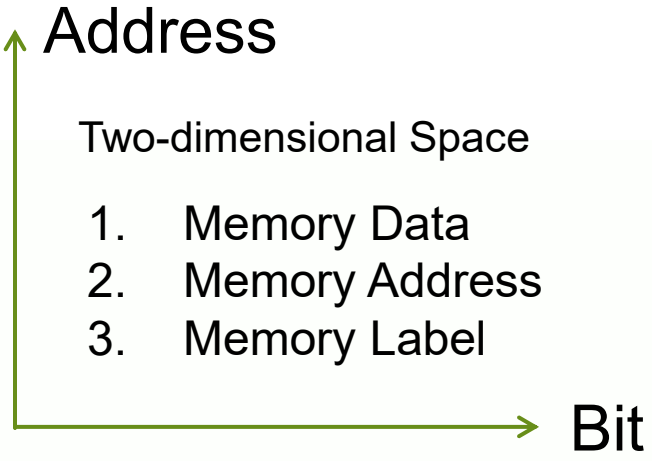


- 32-bit addresses support 4 GiB memory space
- Code, data, and I/O share same memory space
- Data types are bytes, halfwords, and words
- Memory addresses are byte addresses
- Predefined regions have distinct characteristics
  - Executable
  - Device or Strongly-ordered
  - Shareable

Analogy:
1. Main Building with Guest Rooms.
2. Annex Buildings with Functional Rooms.

On-chip Memory Sub-space in ARM

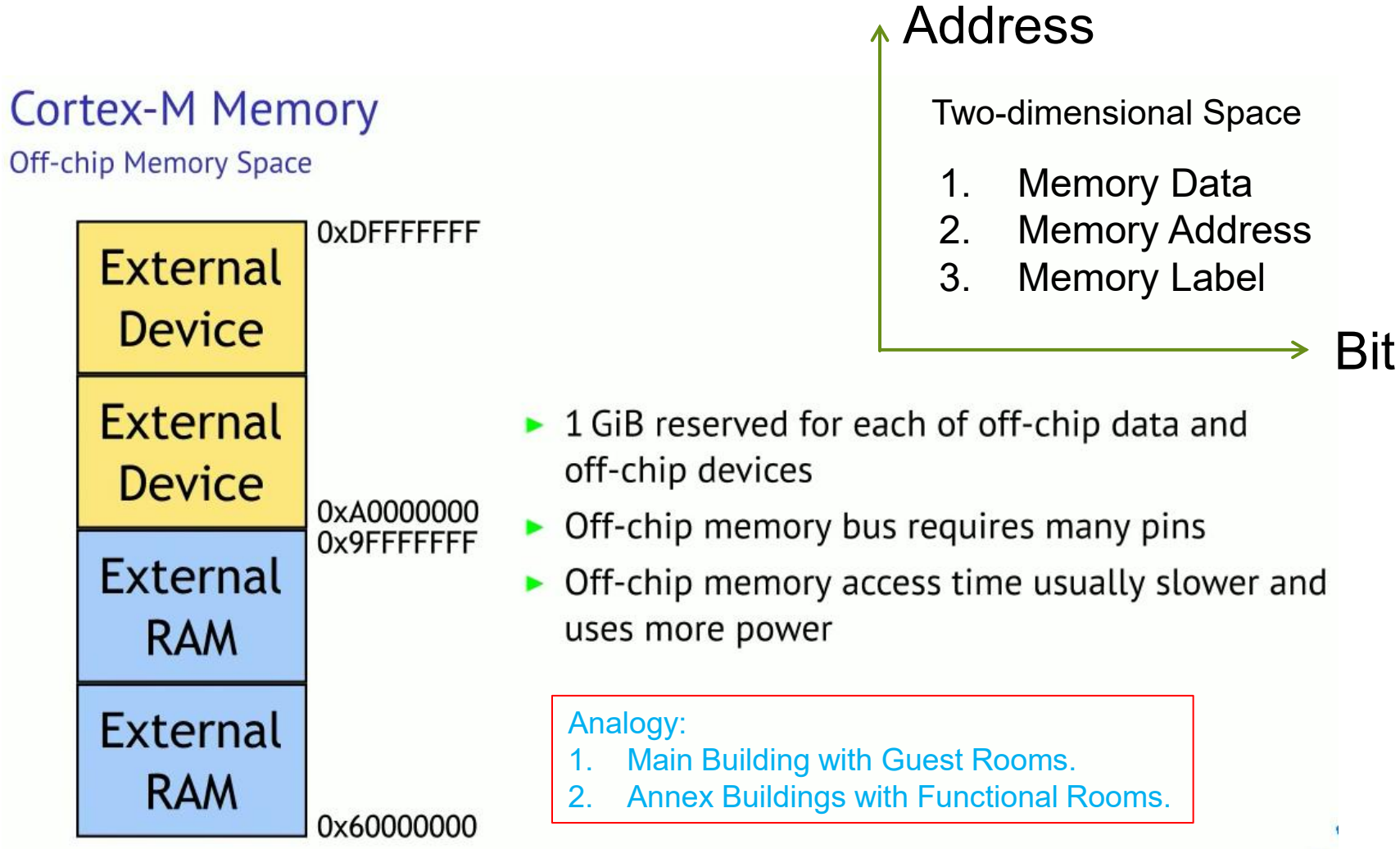




- On-chip code, data, and I/O are located in the first 1.5 GiB of memory space
- Each is allocated 0.5 GiB
- May use physically separate buses for each space

Analogy:
1. Main Building with Guest Rooms.
2. Annex Buildings with Functional Rooms.

Off-chip Memory Sub-space in ARM



Private Memory Sub-space in ARM

Address

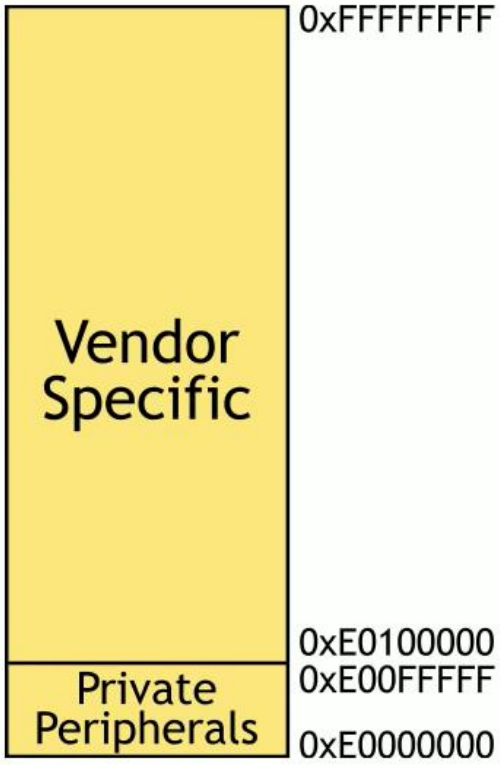
Two-dimensional Space

1. Memory Data
2. Memory Address
3. Memory Label

Bit

Cortex-M Memory

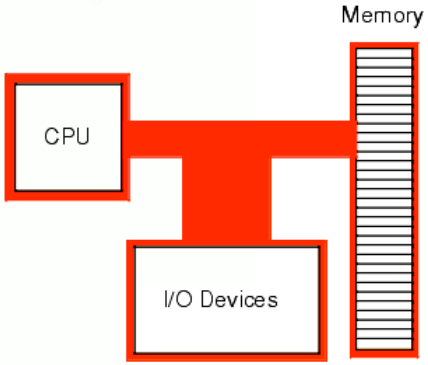
Private Memory Space



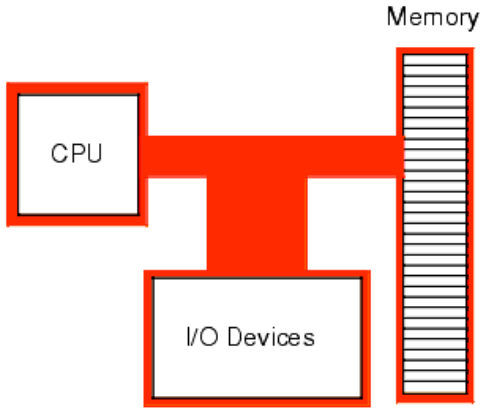
Private Peripheral Bus occupies 1 MiB space

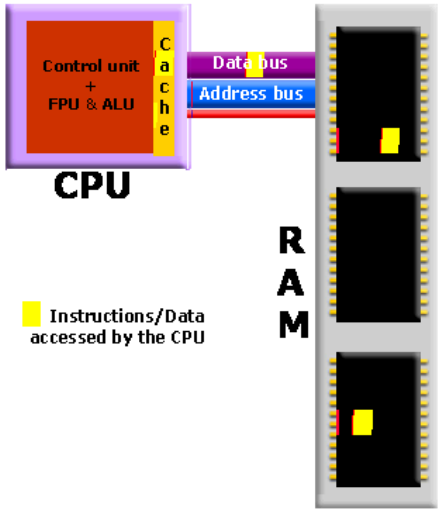
Registers that control peripherals that are a mandatory part of the Cortex-M architecture are mapped here.

- Nested Vectored Interrupt Controller (NVIC)
- System Tick Timer (SysTick)
- Fault status and control
- Processor debugging

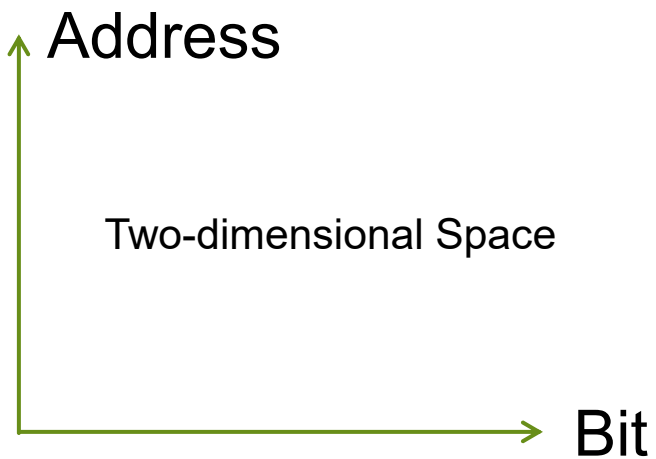


Outline





- Binary Logic Devices
- Memory Construction
- Memory Space in ARM
- Memory-Centric Operations
- Memory-Mapped I/O Devices



1. Memory Content
2. Memory Address
3. Memory Label

Types of Values inside Memory

Values of Data/Codes

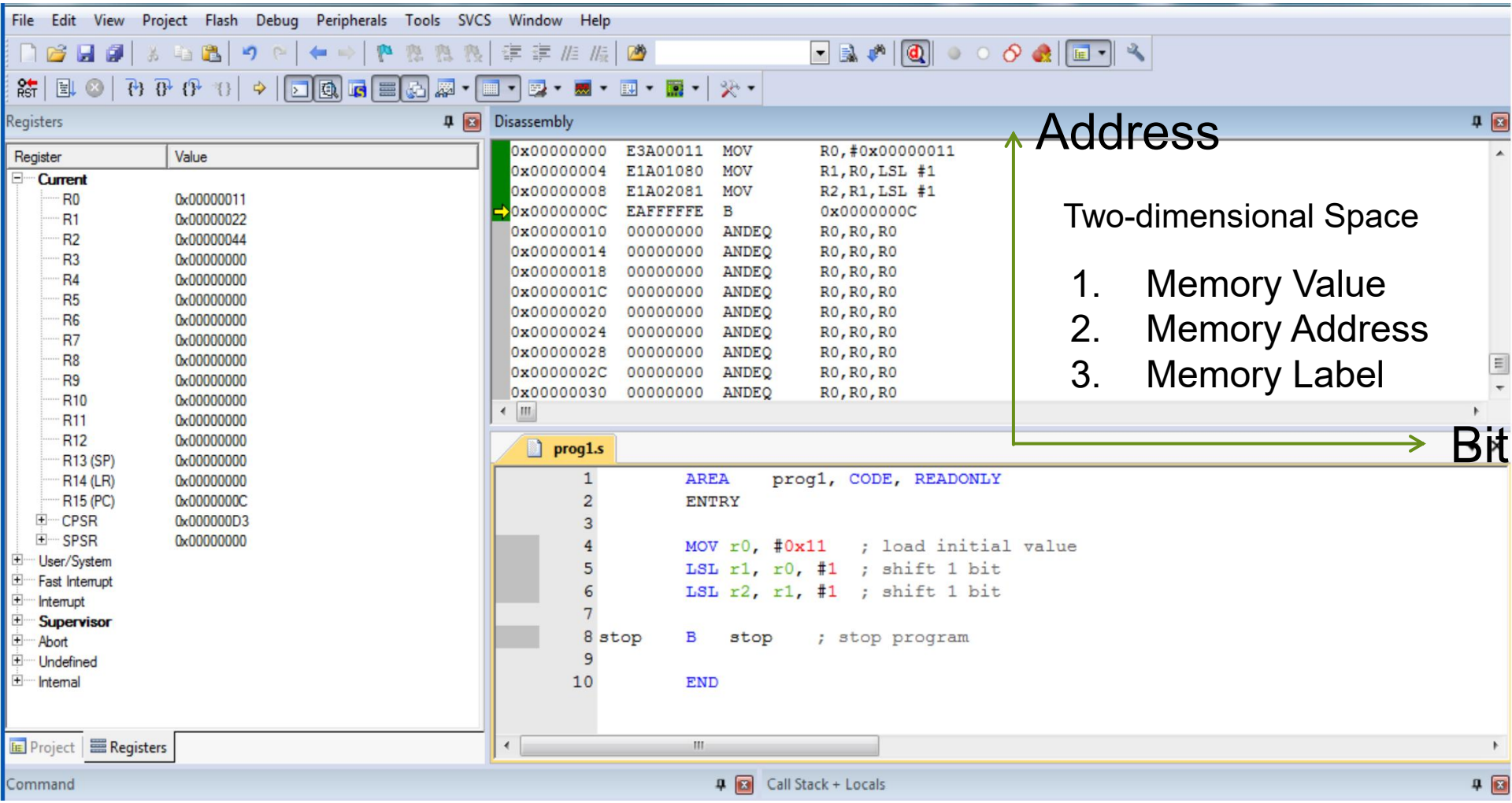
- int x, y, z ;
- x = 10.0 ;
- y = 5.0 ;
- z = x + y ;

Values of Addresses

- int *x, *y, *z, a, b, c ;
- x = &a; *x = a;
- y = &b; *y = b;
- z = &c ; *z = c;

An *ampersand* is a logogram "*&*" representing the conjunction word "and".
In C, it means the address of a variable.

Example


File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Register Value
Current
R0 0x00000011
R1 0x00000022
R2 0x00000044
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x00000000
R14 (LR) 0x00000000
R15 (PC) 0x0000000C
CPSR 0x000000D3
SPSR 0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
Project Registers
Disassembly
0x00000000 E3A00011 MOV R0,#0x00000011
0x00000004 E1A01080 MOV R1,R0,LSL #1
0x00000008 E1A02081 MOV R2,R1,LSL #1
0x0000000C EAFFFFFE B 0x0000000C
0x00000010 00000000 ANDEQ R0,R0,R0
0x00000014 00000000 ANDEQ R0,R0,R0
0x00000018 00000000 ANDEQ R0,R0,R0
0x0000001C 00000000 ANDEQ R0,R0,R0
0x00000020 00000000 ANDEQ R0,R0,R0
0x00000024 00000000 ANDEQ R0,R0,R0
0x00000028 00000000 ANDEQ R0,R0,R0
0x0000002C 00000000 ANDEQ R0,R0,R0
0x00000030 00000000 ANDEQ R0,R0,R0
Address
Two-dimensional Space
1. Memory Value
2. Memory Address
3. Memory Label
Bit
prog1.s
1 AREA prog1, CODE, READONLY
2 ENTRY
3
4 MOV r0, #0x11 ; load initial value
5 LSL r1, r0, #1 ; shift 1 bit
6 LSL r2, r1, #1 ; shift 1 bit
7
8 stop B stop ; stop program
9
10 END
Command
Call Stack + Locals

ARM's Instruction Format

Opcodes & Operands

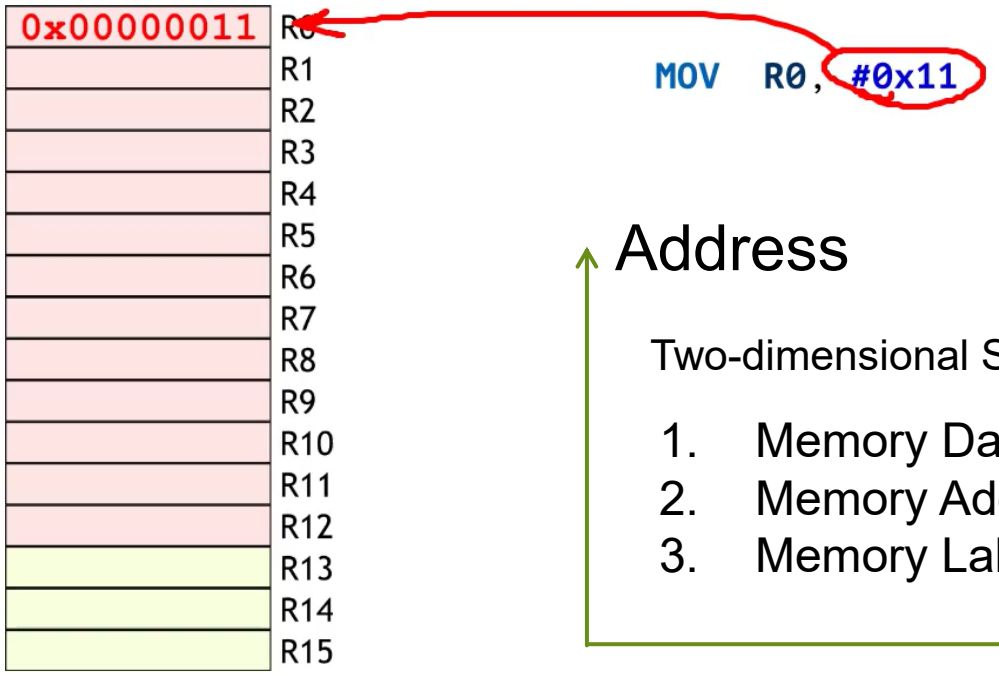
Instructions comprises; 4-byte instruction; of Opcode and Operands

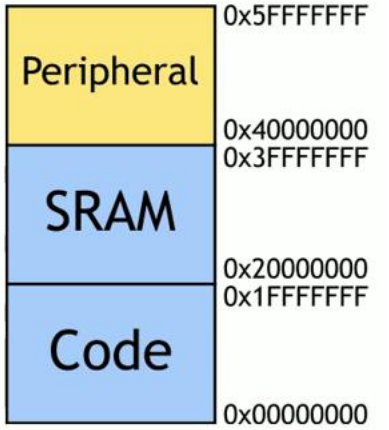
31 30 29 28	27 26 25	24 23 22 21	20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0	
Cond	0 0 1	Opcode	S	Rn	Rd	Operand2			Data processing/ PSR Transfer
Cond	0 0 0	0 0 0 A	S	Rd	Rn	Rs	1 0 0 1	Rm	Multiply
Cond	0 0 0	0 1 U A	S	RdHi	RnLo	Rn	1 0 0 1	Rm	Multiply Long
Cond	0 0 0	1 0 B 0	0	Rn	Rd	0 0 0 0	1 0 0 1	Rm	Single data swap
Cond	0 0 0	1 0 0 1	0	1 1 1 1	1 1 1 1	1 1 1 1	0 0 0 1	Rn	Branch and exchange
Cond	0 0 0	P U 0 W	L	Rn	Rd	0 0 0 0	1 S H 1	Rm	Halfword data transfer: register offset
Cond	0 0 0	P U 1 W	L	Rn	Rd	Offset	1 S H 1	Offset	Halfword data transfer: immediate offset
Cond	0 1 1	P U B W	L	Rn	Rd	Offset			Single data transfer
Cond	0 1 1						1		Undefined
Cond	1 0 0	P U S W	L	Rn	Register List				Block data transfer
Cond	1 0 1	L	Offset						Branch
Cond	1 1 0	P U N W	L	Rn	CRd	CP#	Offset		Coprocessor data transfer
Cond	1 1 1	0	CP Opc	CRn	CRd	CP#	CP#	0 CRm	Coprocessor data Operation
Cond	1 1 1	0	CP Opc	L	CRn	Rd	CP#	CP#	1 CRm	Coprocessor register Transfer
Cond	1 1 1	1	Ignored by processor						Software Interrupt

ARM's Instruction Format (continued)

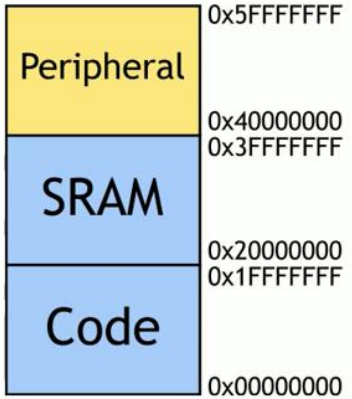
- Format 1: Opcode DestReg, Operand2
- Format 2: Opcode DestReg, SrcReg, Operand2

- Example: The Move Instruction





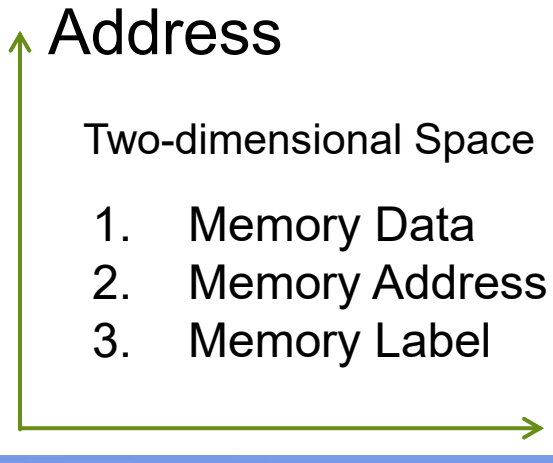
Example of Move Data inside ARM



The Move Instruction

0x00000011	R0
0x00000A00	R1
0xFFFFFFFB	R2
0xFEEDC0DE	R3
0x00000A00	R4
	R5
	R6
	R7
	R8
	R9
	R10
	R11
	R12
	R13
	R14
	R15

```
MOV  R0, #0x11
MOV  R1, #2560
MVN  R2, #4
MOVW R3, #0xC0DE
MOVT R3, #0xFEED
MOV  R4, R1
```



Two-dimensional Space

1. Memory Data
2. Memory Address
3. Memory Label

Example of Load Operation in ARM

Basic Load/Store Instructions

1. Use the memory value as an address
2. Get the value from the address

0x00000011	R0
0x00000A00	R1
0xFFFFFFFB	R2
0xFEEDC0DE	R3
0x00000A00	R4
	R5
	R6
	R7
	R8
	R9
	R10
	R11
	R12
	R13
	R14
	R15

```
LDR   R5, [R1]
```

0xAAAAAAAA	0x000009FC
0x0000BEAD	0x00000A00
0x55555555	0x00000A04

R5 = 0x0000BEAD

Address

Two-dimensional Space

1. Memory Data
2. Memory Address
3. Memory Label

Bit

Example of Store Operation in ARM

Basic Load/Store Instructions

1. Use the memory value as an address
2. Get the value from the address

0x00000011	R0
0x00000A00	R1
0xFFFFFFFB	R2
0xFEEDC0DE	R3
0x00000A00	R4
0x0000BEAD	R5
	R6
	R7
	R8
	R9
	R10
	R11
	R12
	R13
	R14
	R15

```
LDR   R5, [R1]
STR   R3, [R1]
```

0xAAAAAAAA	0x000009FC
0xFEEDCODE	0x00000A00
0x55555555	0x00000A04

*0x00000A00 = 0xFEEDC0DE

Address

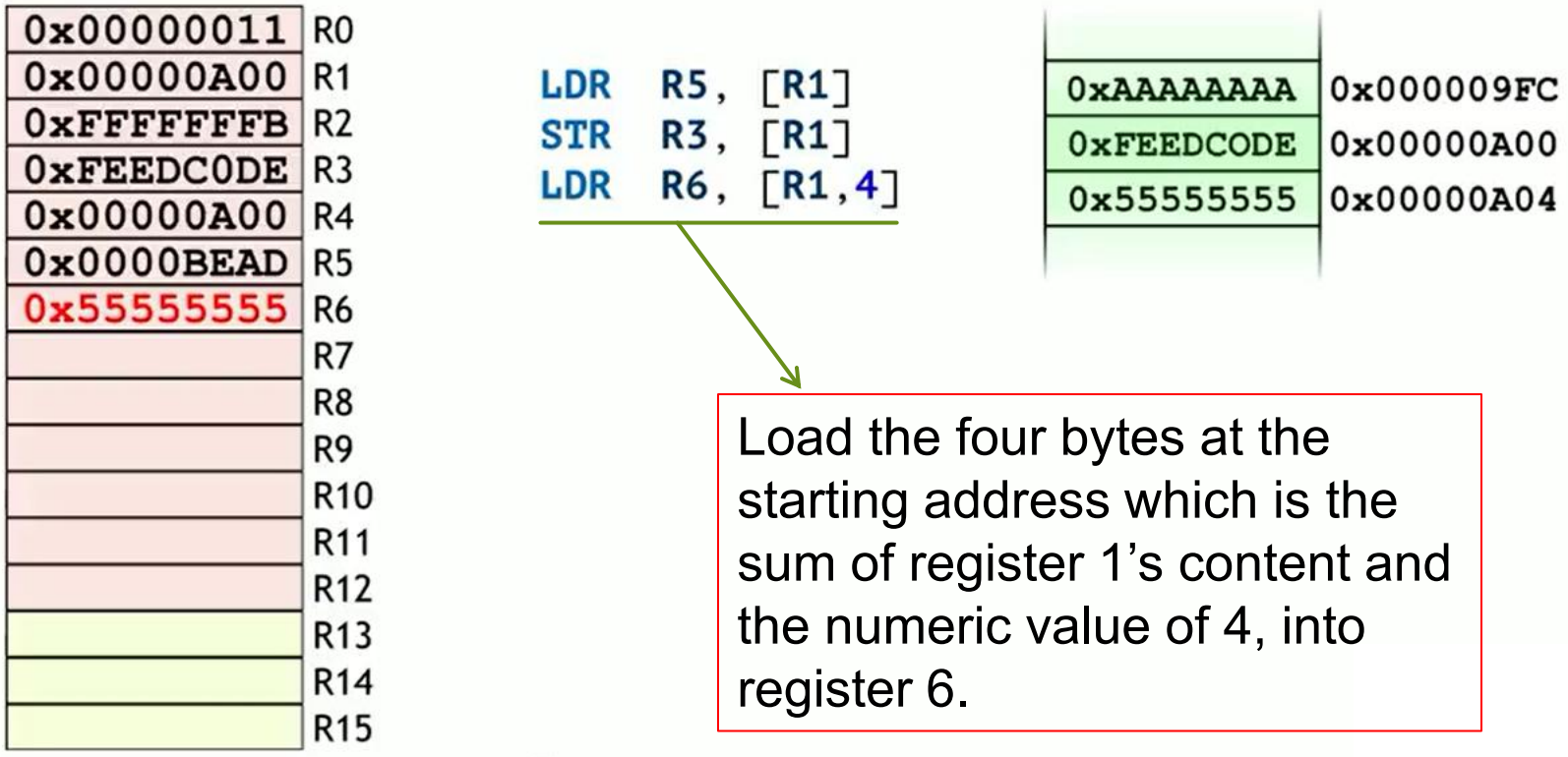
Two-dimensional Space

1. Memory Data
2. Memory Address
3. Memory Label

Bit

More Example

Explain the meaning of the last instruction.



Example of Addition in ARM

Arithmetic Operators

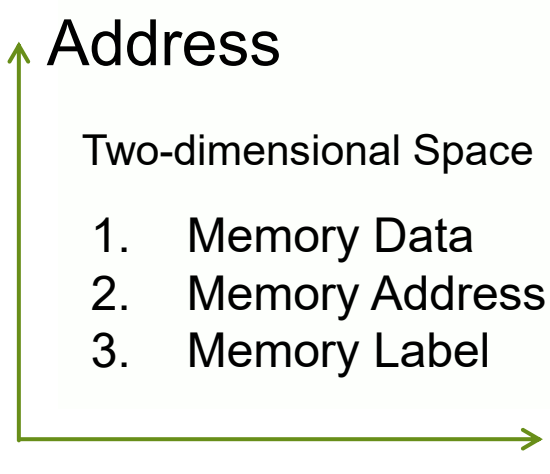
Addition

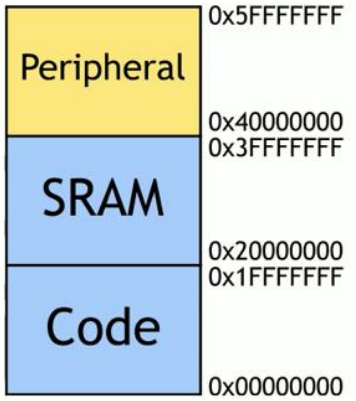
In assembly we write:

```
ADD   R1, R0, R0
ADD   R2, R0, #2
```

0x00000002	R0
0x00000004	R1
0x00000004	R2
0x40000000	R3
0x60000000	R4
	R5
	R6
	R7
	R8
	R9
	R10
	R11
	R12
	R13
	R14
	R15

```
  2
+ 2
---
  4
```





Example of Multiplication in ARM

Arithmetic Operators

Multiplication

In assembly we write:

```
MUL   R2, R0, R1
MUL   R5, R4, R3
MUL   R8, R6, R7
```

0x00000002	R0
0x00000004	R1
0x00000008	R2
0xFFFFFF10	R3
0x00000077	R4
0xFFFF9070	R5
0x0000BEAD	R6
0x000157B5	R7
0x00009B51	R8
	R9
	R10
	R11
	R12
	R13
	R14
	R15

(there is an overflow)

With overflow

$$\begin{array}{r} 48813 \\ \times\ 87989 \\ \hline 39761 \end{array}$$

True Result

$$\begin{array}{r} 48813 \\ \times\ 87989 \\ \hline 4295007057 \end{array}$$

0xBEAD = 48813
0x157B5 = 87989
0x9B51 = 39761
0x100009B51 = 4295007057

Example of Division in ARM

Arithmetic Operators

Division

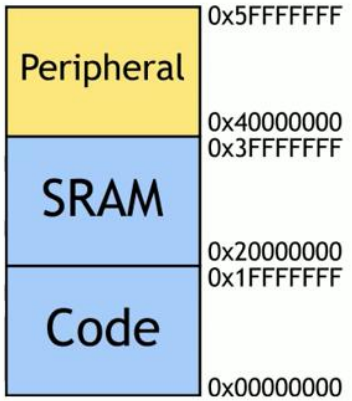
In assembly we write:

```
UDIV R3, R2, R0
UDIV R4, R2, R1
UDIV R5, R1, R2
```

0x00000002	R0
0x00000003	R1
0x00000004	R2
0x00000002	R3
0x00000001	R4
0x00000000	R5
0xFFFFFF00	R6
0x00000005	R7
	R8
	R9
	R10
	R11
	R12
	R13
	R14
	R15

3 / 4 = 0

(zero result)



Example of Forcing Bits to 1 in ARM

R0 = 0000 0001 0001 0000

R1 = 0000 0000 0001 0001

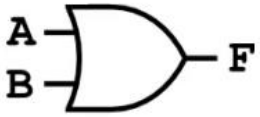
R3 = 1010 1011 1100 1101

R2 = 0000 0001 0001 0001

R4 = 1010 1011 1101 1101

Bit Banging

Forcing bits to 1



A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

In C we would write:

```
# define MASK 0x0110
   DestA = SrcA | MASK;
   DestB = SrcB | MASK;
```

In assembly we write:

```
MOVW R0, #0x0110
ORR  R2, R1, R0
ORR  R4, R3, R0
```

0x00000110	R0
0x00000011	R1
0x00000111	R2
0x0000ABCD	R3
0x0000ABDD	R4
	R5
	R6
	R7
	R8
	R9
	R10
	R11
	R12
	R13
	R14
	R15

Binary Numbers: a series of bits
Hexadecimal Numbers: a series of half-bytes

Example of Forcing Bits to 0 in ARM

Bit Banging

Forcing bits to 0

R0 = 0000 0001 0001 0000

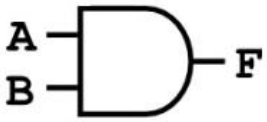
~R0 = 1111 1110 1110 1111

R1 = 0000 0000 0001 0001

R3 = 1010 1011 1100 1101

R2 = 0000 0000 0000 0001

R4 = 1010 1010 1100 1101



A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

In C we would write:

```
# define MASK 0x0110
   DestA = SrcA & ~MASK;
   DestB = SrcB & ~MASK;
```

In assembly we write:

```
MOVW R0, #0x0110
MVN  R0, R0
AND  R2, R1, R0
AND  R4, R3, R0
```

0xFFFFFEEF	R0
0x00000011	R1
0x00000001	R2
0x0000ABCD	R3
0x0000AACD	R4
	R5
	R6
	R7
	R8
	R9
	R10
	R11
	R12
	R13
	R14
	R15

Binary Numbers: a series of bits
Hexadecimal Numbers: a series of half-bytes

Example of Flipping Bits in ARM

Bit Banging

Flipping bits

R0 = 0000 0001 0001 0000

R1 = 0000 0000 0001 0001

R3 = 1010 1011 1100 1101

R2 = 0000 0001 0000 0001

R4 = 1010 1010 1101 1101

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

In C we would write:

```
# define MASK 0x0110
   DestA = SrcA ^ MASK;
   DestB = SrcB ^ MASK;
```

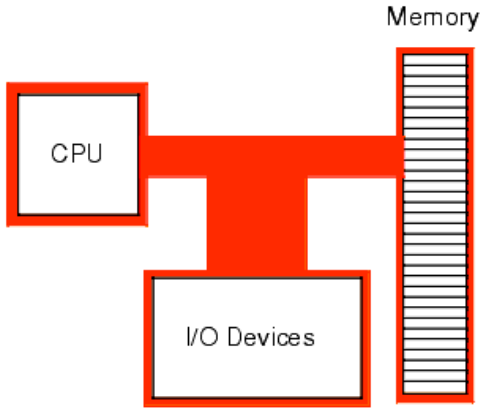
In assembly we write:

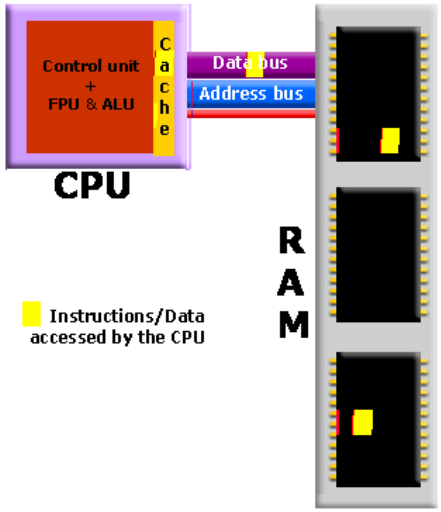
```
MOVW R0, #0x0110
EOR  R2, R1, R0
EOR  R4, R3, R0
```

0x00000110	R0
0x00000011	R1
0x00000101	R2
0x0000ABCD	R3
0x0000AADD	R4
	R5
	R6
	R7
	R8
	R9
	R10
	R11
	R12
	R13
	R14
	R15

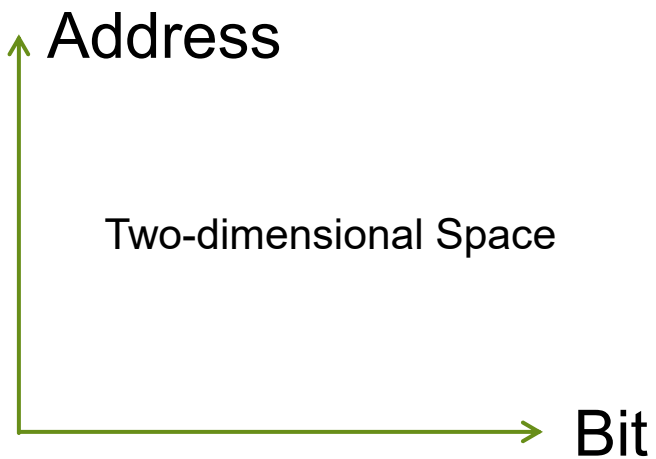
Binary Numbers: a series of bits
Hexadecimal Numbers: a series of half-bytes

Outline



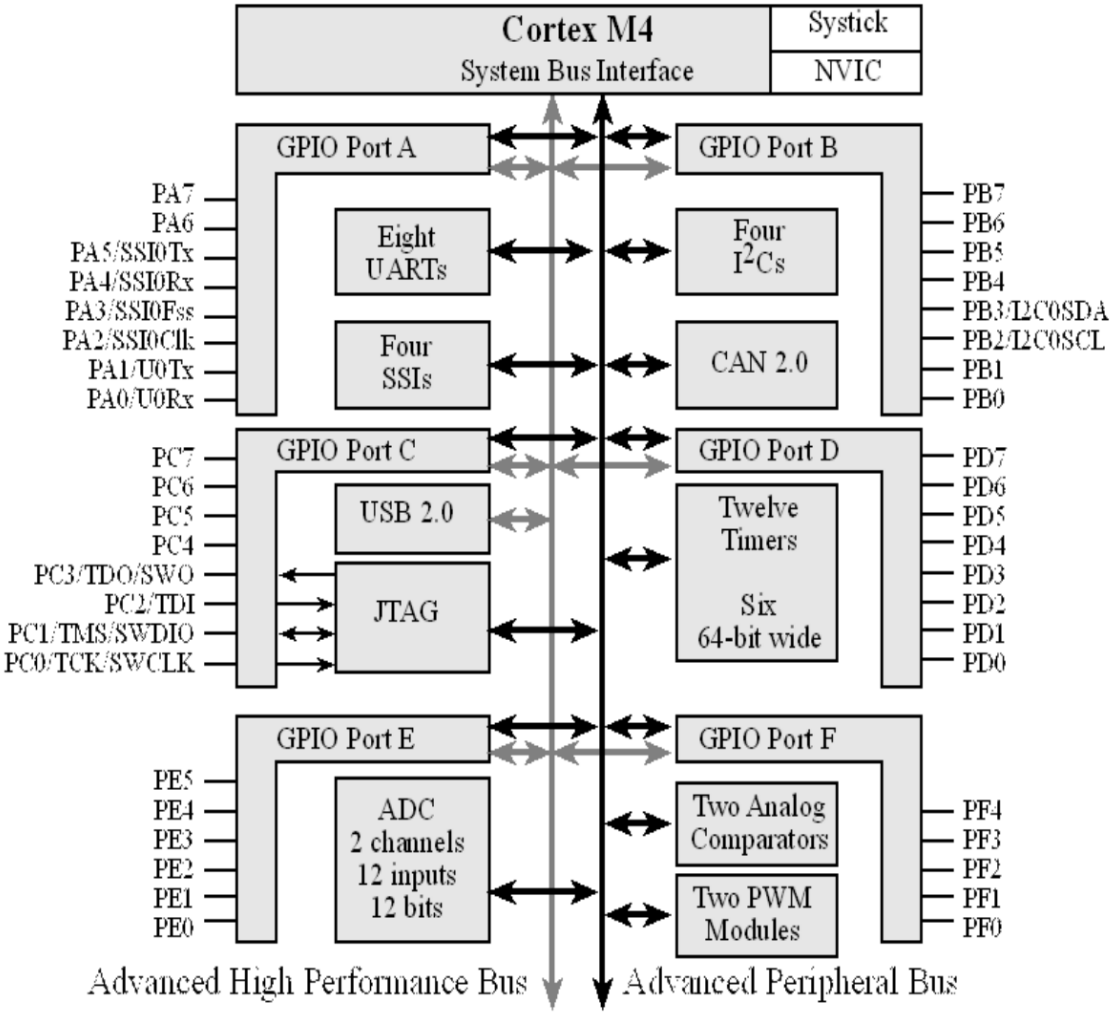


- Binary Logic Devices
- Memory Construction
- Memory Space in ARM
- Memory-Centric Operations
- Memory-Mapped I/O Devices

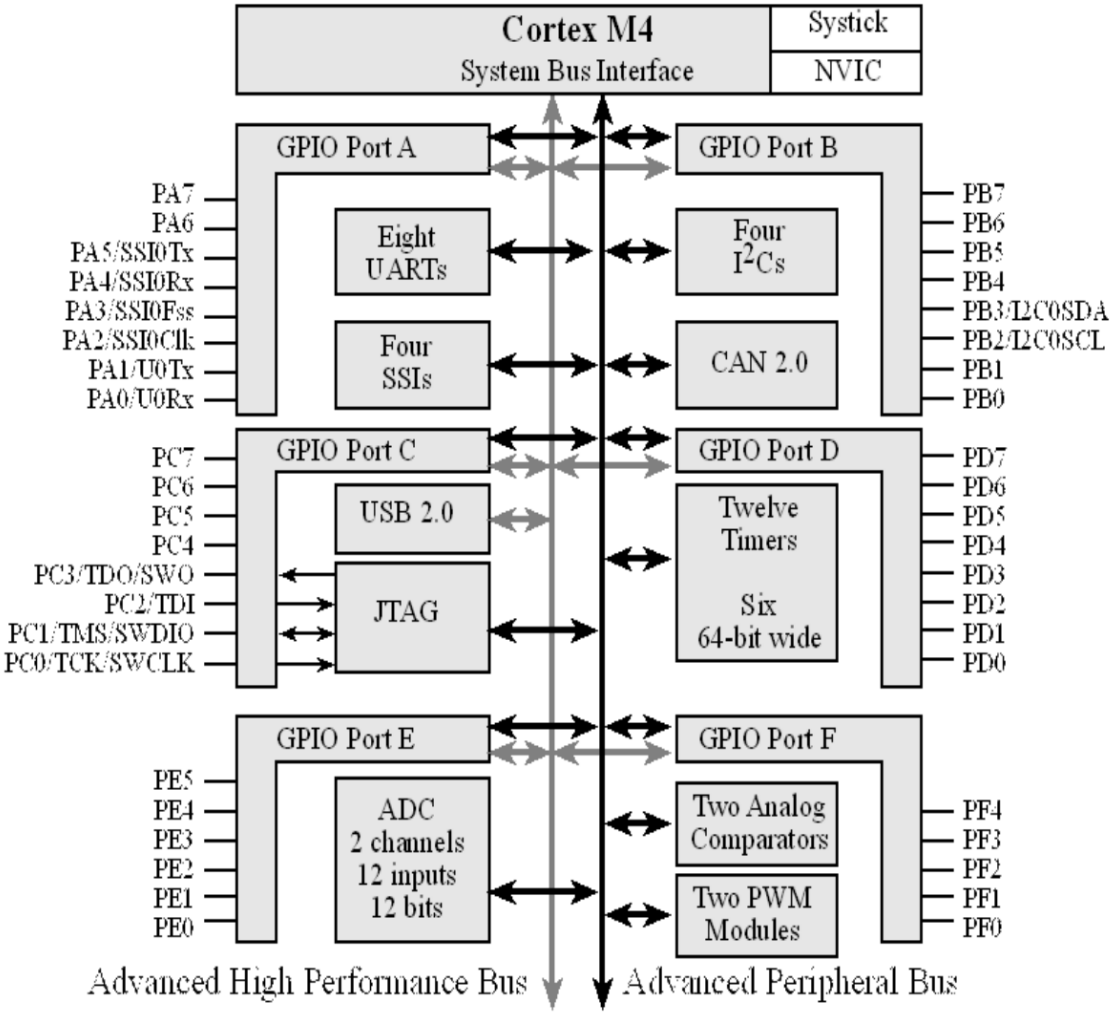


1. Memory Content
2. Memory Address
3. Memory Label

Each I/O Module Has a Port …

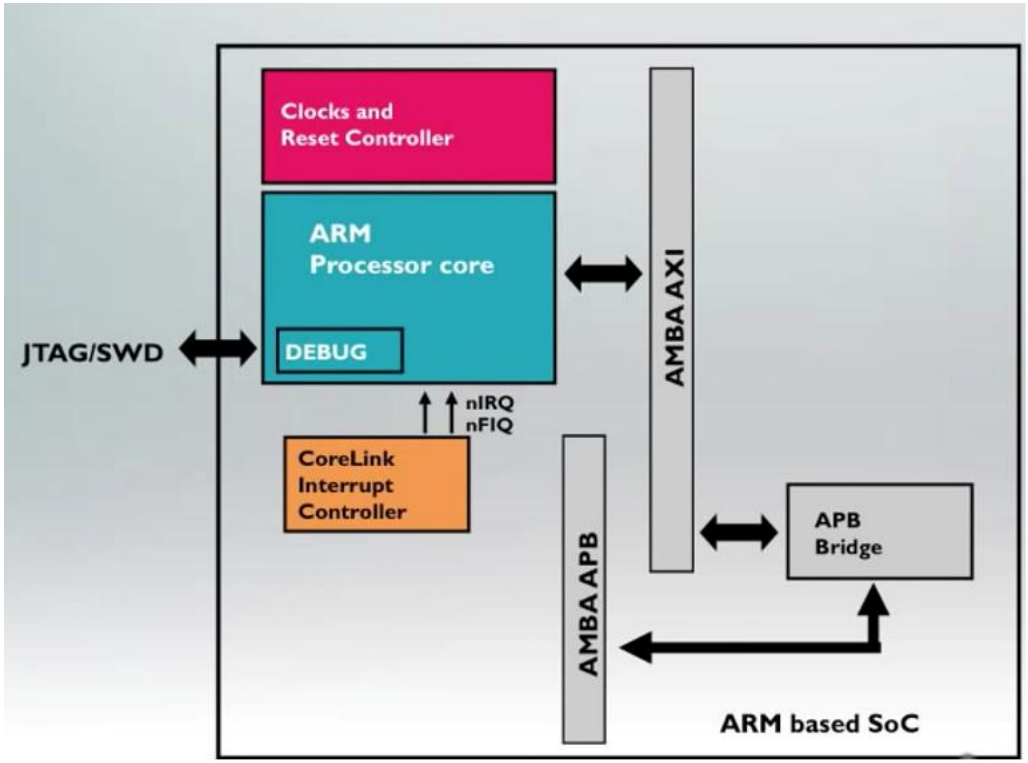


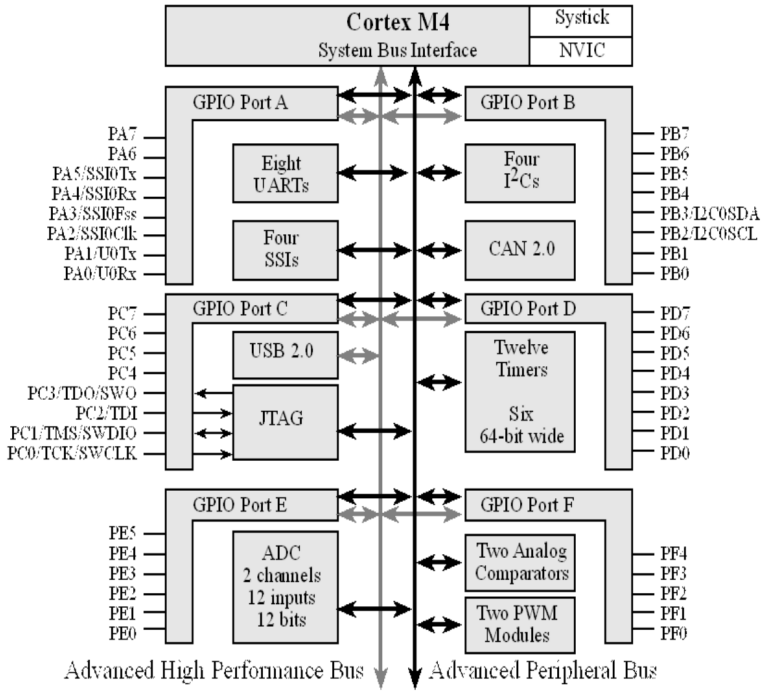
All Ports Are Connected by Data Buse …



ARM Cortex has two internal buses

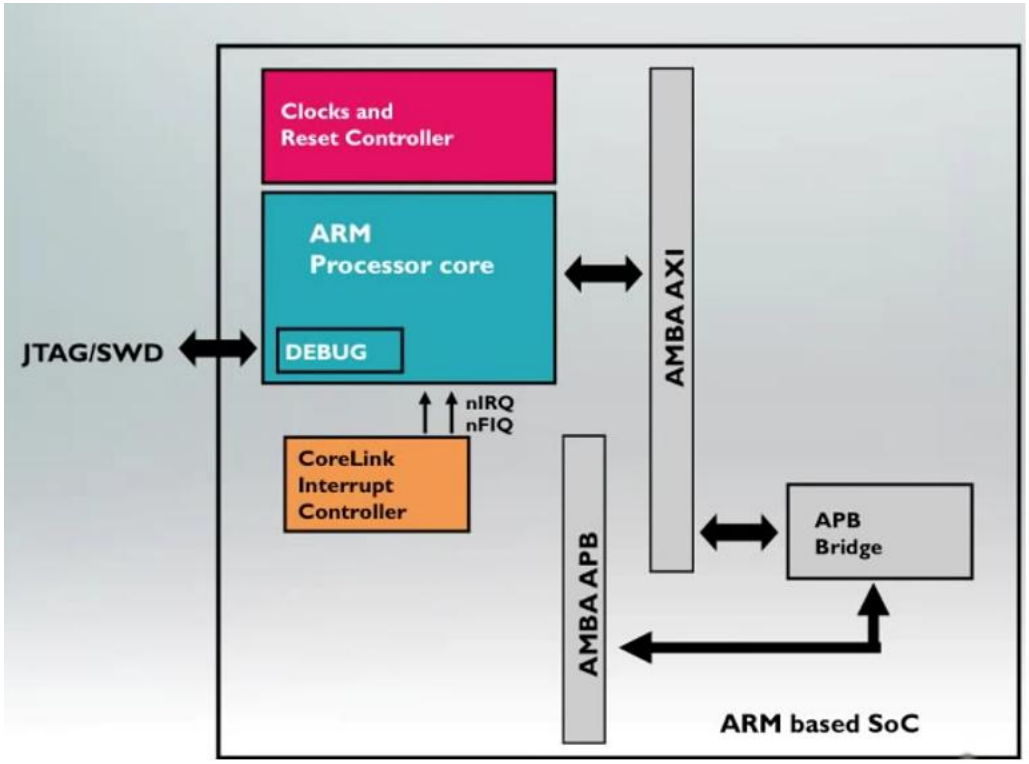
- The ARM **Advanced Microcontroller Bus Architecture** (**AMBA**) is an open-standard, on-chip interconnect specification for the connection and management of functional blocks in system-on-a-chip (SoC) designs. It facilitates development of multi-processor designs with large numbers of controllers and peripherals.

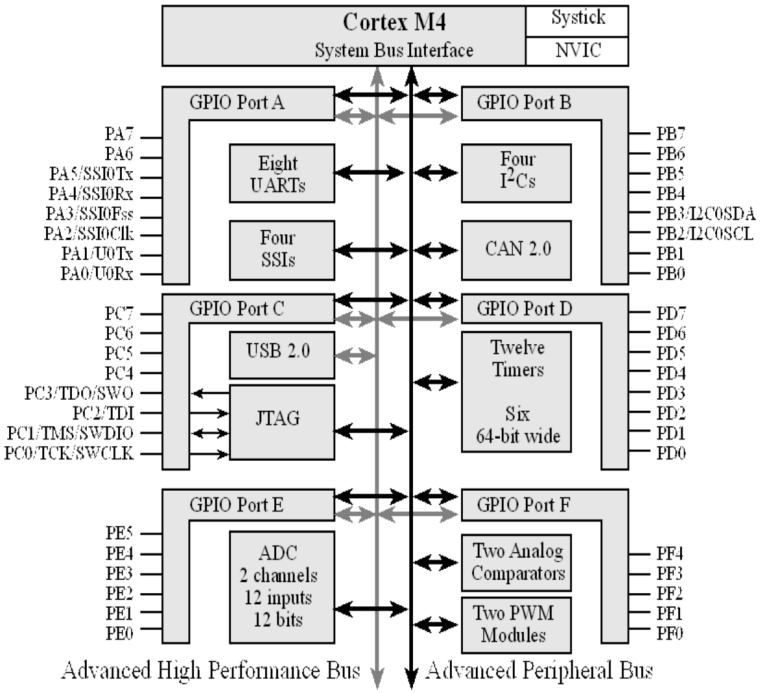




ARM Cortex supports Fast Interrupt Request

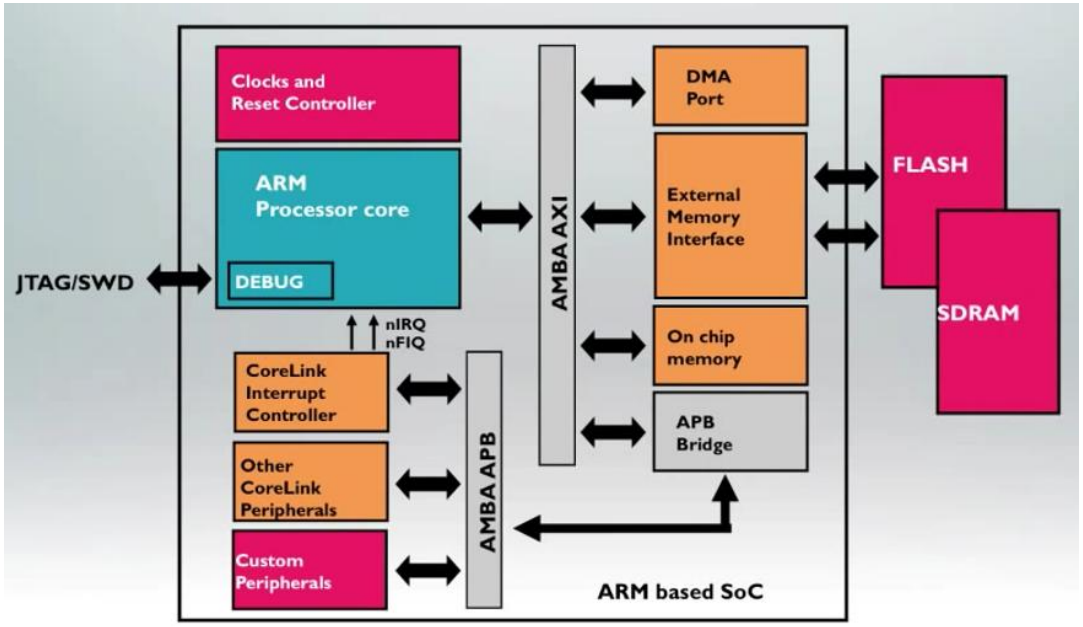
- An **FIQ** is just a higher priority interrupt request, that is prioritized by disabling IRQ and other **FIQ** handlers during request servicing. Therefore, no other interrupts can occur during the processing of the active **FIQ** interrupt.

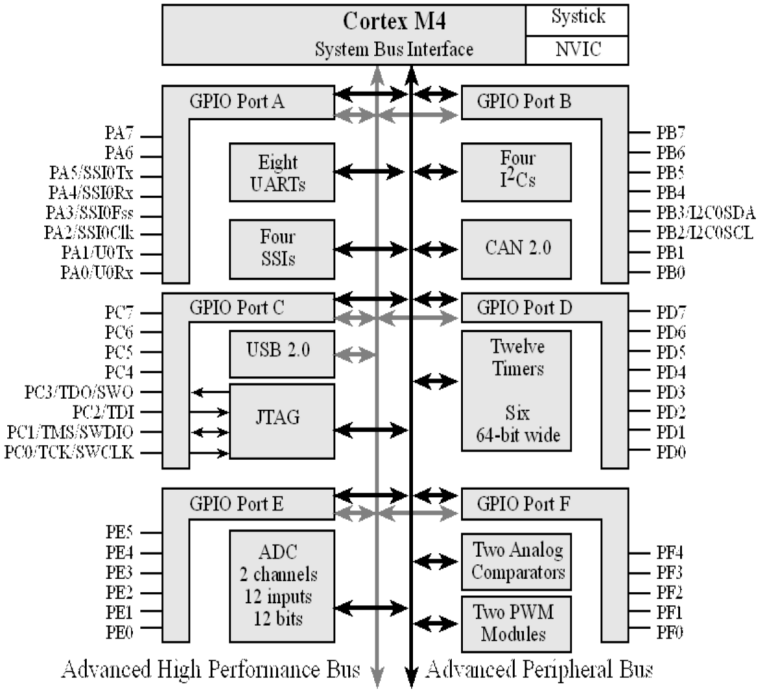




ARM Cortex supports Direct Memory Access

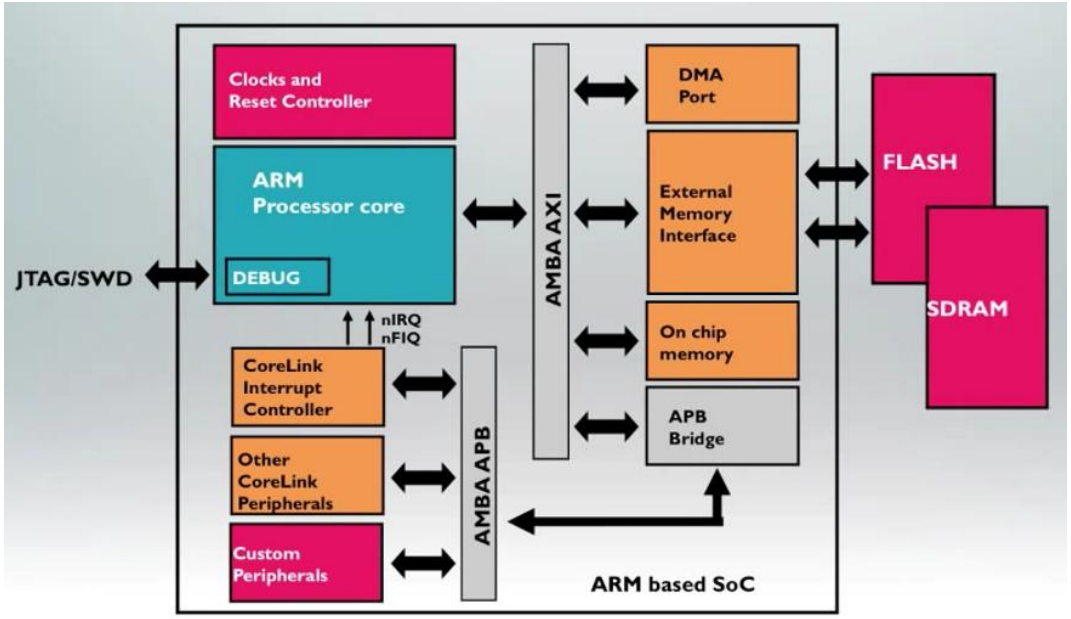
- Direct memory access (**DMA**) is a feature of microprocessor systems that allows certain hardware subsystems to access main system memory such as RAM (Random Access Memory) independently of the central processing unit (CPU).

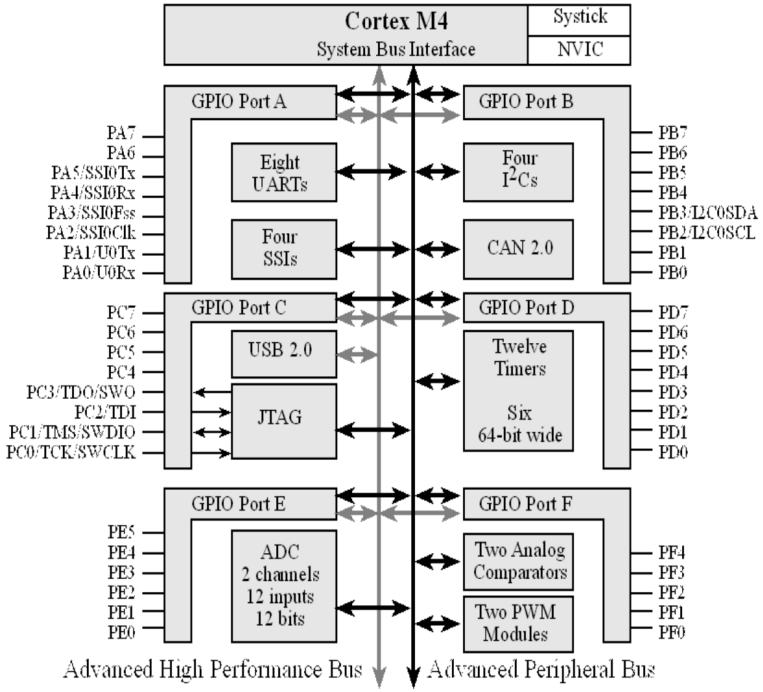




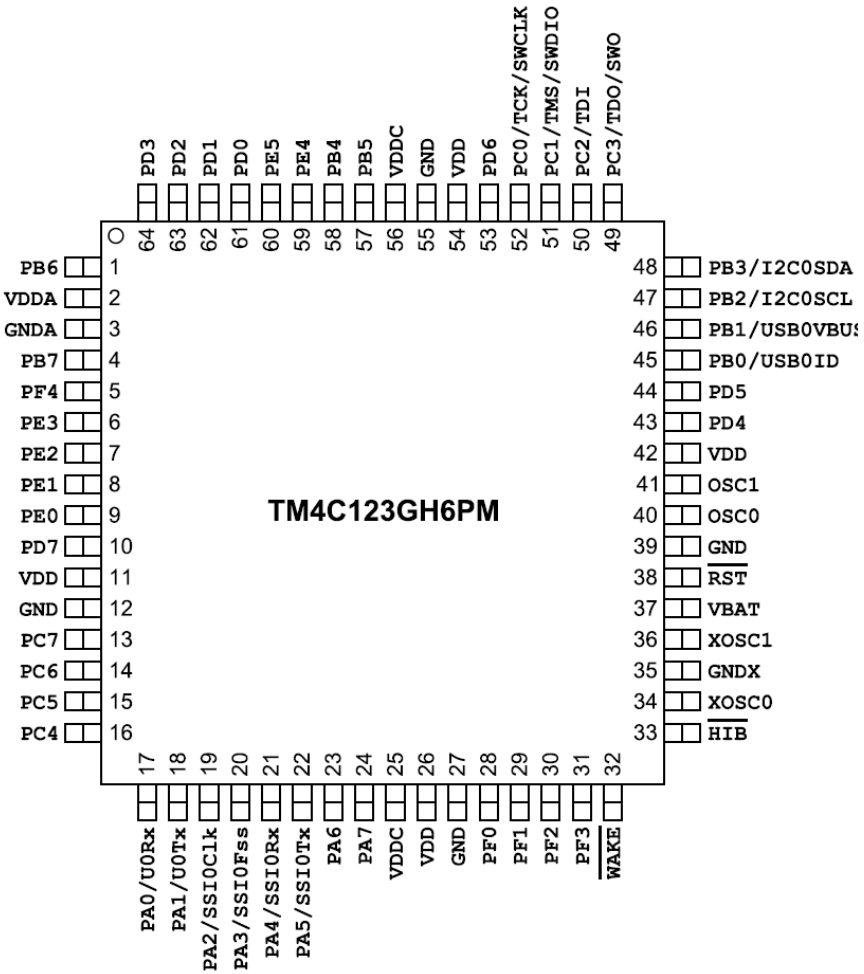
ARM Cortex includes Debug Port

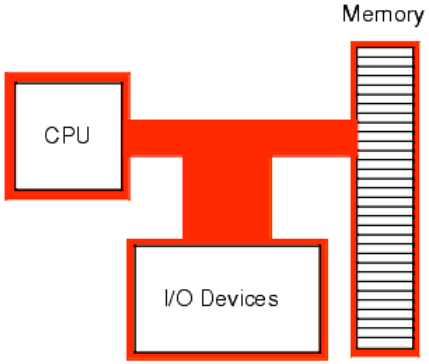
- JTAG (Joint Test Action Group) specifies the use of a dedicated debug port implementing a serial communications interface for low-overhead access without requiring direct external access to the system address and data buses.





Cortex M4's Pin Diagram





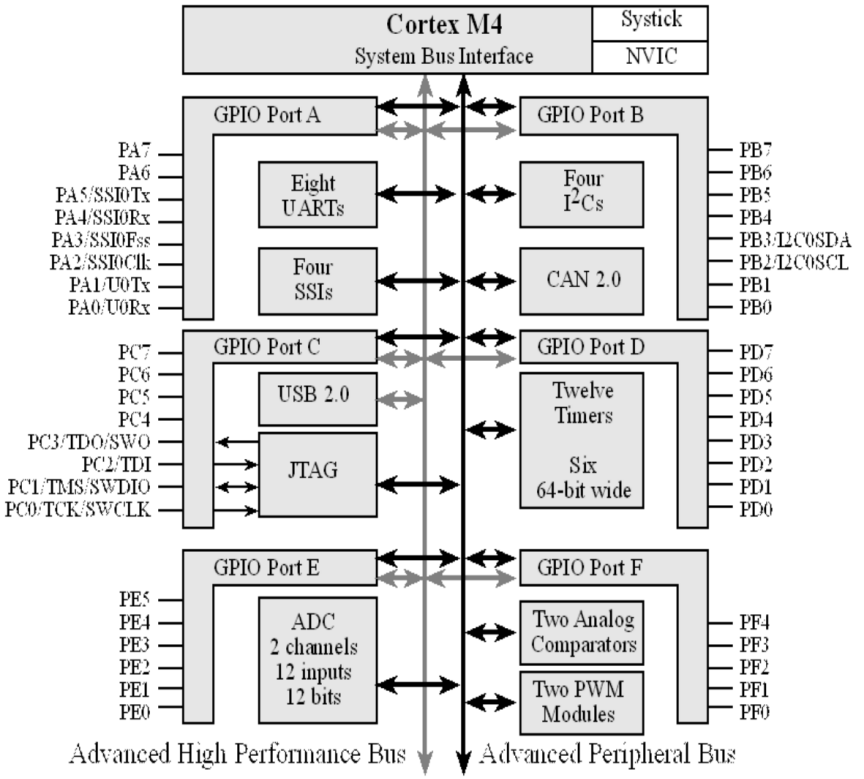


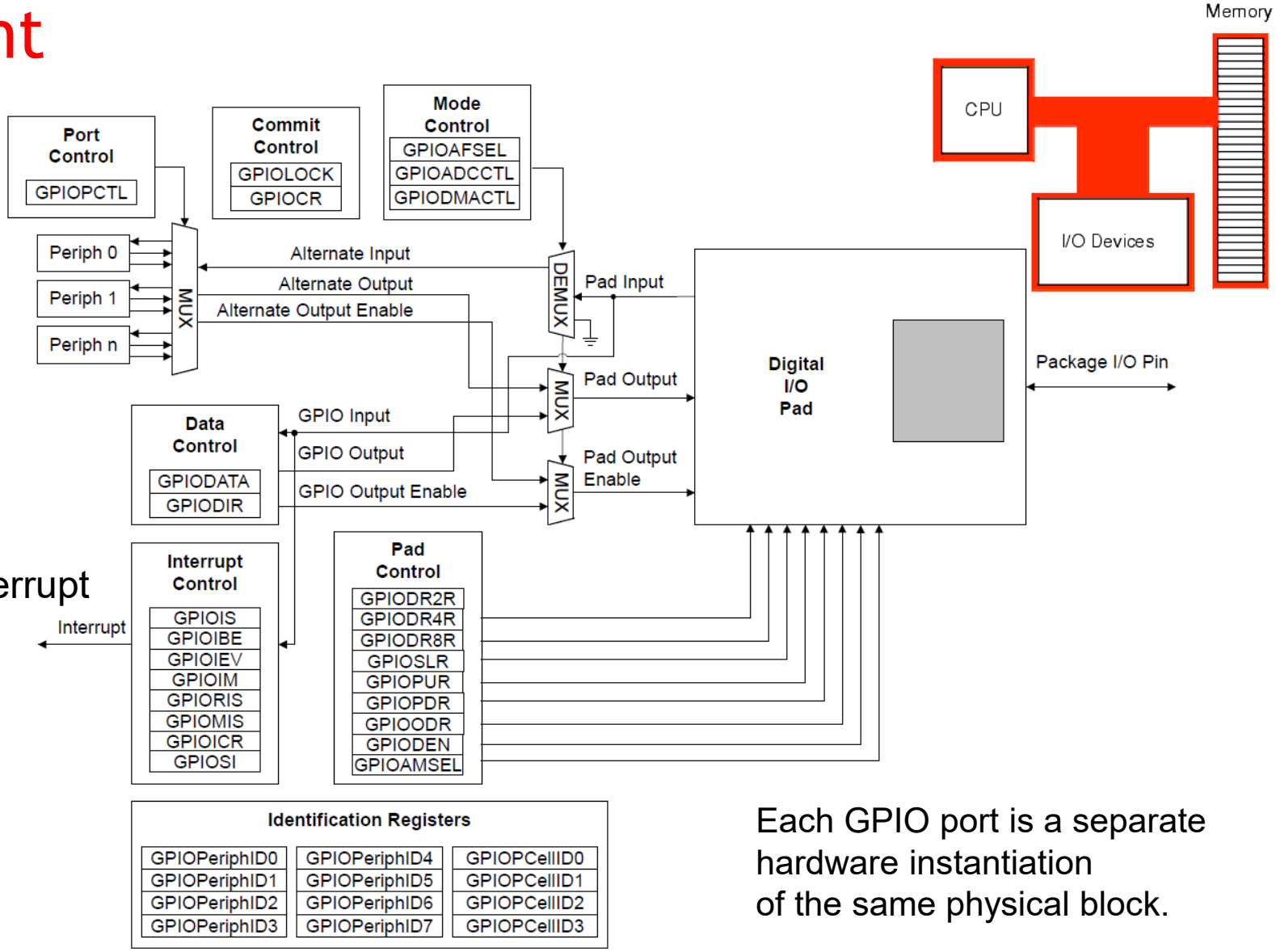
Table of Pins' Usages

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	14	15
PA0	17	-	U0Rx	-	-	-	-	-	-	CAN1Rx	-	-	-
PA1	18	-	U0Tx	-	-	-	-	-	-	CAN1Tx	-	-	-
PA2	19	-	-	SSI0Clk	-	-	-	-	-	-	-	-	-
PA3	20	-	-	SSI0Fss	-	-	-	-	-	-	-	-	-
PA4	21	-	-	SSI0Rx	-	-	-	-	-	-	-	-	-
PA5	22	-	-	SSI0Tx	-	-	-	-	-	-	-	-	-
PA6	23	-	-	-	I2C1SCL	-	M1PWM2	-	-	-	-	-	-
PA7	24	-	-	-	I2C1SDA	-	M1PWM3	-	-	-	-	-	-
PB0	45	USB0ID	U1Rx	-	-	-	-	-	T2CCP0	-	-	-	-
PB1	46	USB0VBUS	U1Tx	-	-	-	-	-	T2CCP1	-	-	-	-
PB2	47	-	-	-	I2C0SCL	-	-	-	T3CCP0	-	-	-	-
PB3	48	-	-	-	I2C0SDA	-	-	-	T3CCP1	-	-	-	-
PB4	58	AIN10	-	SSI2Clk	-	M0PWM2	-	-	T1CCP0	CAN0Rx	-	-	-
PB5	57	AIN11	-	SSI2Fss	-	M0PWM3	-	-	T1CCP1	CAN0Tx	-	-	-
PB6	1	-	-	SSI2Rx	-	M0PWM0	-	-	T0CCP0	-	-	-	-
PB7	4	-	-	SSI2Tx	-	M0PWM1	-	-	T0CCP1	-	-	-	-

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	14	15
PC0	52	-	TCK SWCLK	-	-	-	-	-	T4CCP0	-	-	-	-
PC1	51	-	TMS SWDIO	-	-	-	-	-	T4CCP1	-	-	-	-
PC2	50	-	TDI	-	-	-	-	-	T5CCP0	-	-	-	-
PC3	49	-	TDO SWO	-	-	-	-	-	T5CCP1	-	-	-	-
PC4	16	C1-	U4Rx	U1Rx	-	M0PWM6	-	IDX1	WT0CCP0	U1RTS	-	-	-
PC5	15	C1+	U4Tx	U1Tx	-	M0PWM7	-	PhA1	WT0CCP1	U1CTS	-	-	-
PC6	14	C0+	U3Rx	-	-	-	-	PhB1	WT1CCP0	USB0EPEN	-	-	-
PC7	13	C0-	U3Tx	-	-	-	-	-	WT1CCP1	USB0PFLT	-	-	-
PD0	61	AIN7	SSI3Clk	SSI1Clk	I2C3SCL	M0PWM6	M1PWM0	-	WT2CCP0	-	-	-	-
PD1	62	AIN6	SSI3Fss	SSI1Fss	I2C3SDA	M0PWM7	M1PWM1	-	WT2CCP1	-	-	-	-
PD2	63	AIN5	SSI3Rx	SSI1Rx	-	M0FAULT0	-	-	WT3CCP0	USB0EPEN	-	-	-
PD3	64	AIN4	SSI3Tx	SSI1Tx	-	-	-	IDX0	WT3CCP1	USB0PFLT	-	-	-
PD4	43	USB0DM	U6Rx	-	-	-	-	-	WT4CCP0	-	-	-	-
PD5	44	USB0DP	U6Tx	-	-	-	-	-	WT4CCP1	-	-	-	-
PD6	53	-	U2Rx	-	-	M0FAULT0	-	PhA0	WT5CCP0	-	-	-	-
PD7	10	-	U2Tx	-	-	-	-	PhB0	WT5CCP1	NMI	-	-	-

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	14	15
PE0	9	AIN3	U7Rx	-	-	-	-	-	-	-	-	-	-
PE1	8	AIN2	U7Tx	-	-	-	-	-	-	-	-	-	-
PE2	7	AIN1	-	-	-	-	-	-	-	-	-	-	-
PE3	6	AIN0	-	-	-	-	-	-	-	-	-	-	-
PE4	59	AIN9	U5Rx	-	I2C2SCL	M0PWM4	M1PWM2	-	-	CAN0Rx	-	-	-
PE5	60	AIN8	U5Tx	-	I2C2SDA	M0PWM5	M1PWM3	-	-	CAN0Tx	-	-	-
PF0	28	-	U1RTS	SSI1Rx	CAN0Rx	-	M1PWM4	PhA0	T0CCP0	NMI	C0o	-	-
PF1	29	-	U1CTS	SSI1Tx	-	-	M1PWM5	PhB0	T0CCP1	-	C1o	TRD1	-
PF2	30	-	-	SSI1Clk	-	M0FAULT0	M1PWM6	-	T1CCP0	-	-	TRD0	-
PF3	31	-	-	SSI1Fss	CAN0Tx	-	M1PWM7	-	T1CCP1	-	-	TRCLK	-
PF4	5	-	-	-	-	-	M1FAULT0	IDX0	T2CCP0	USB0EPEN	-	-	-

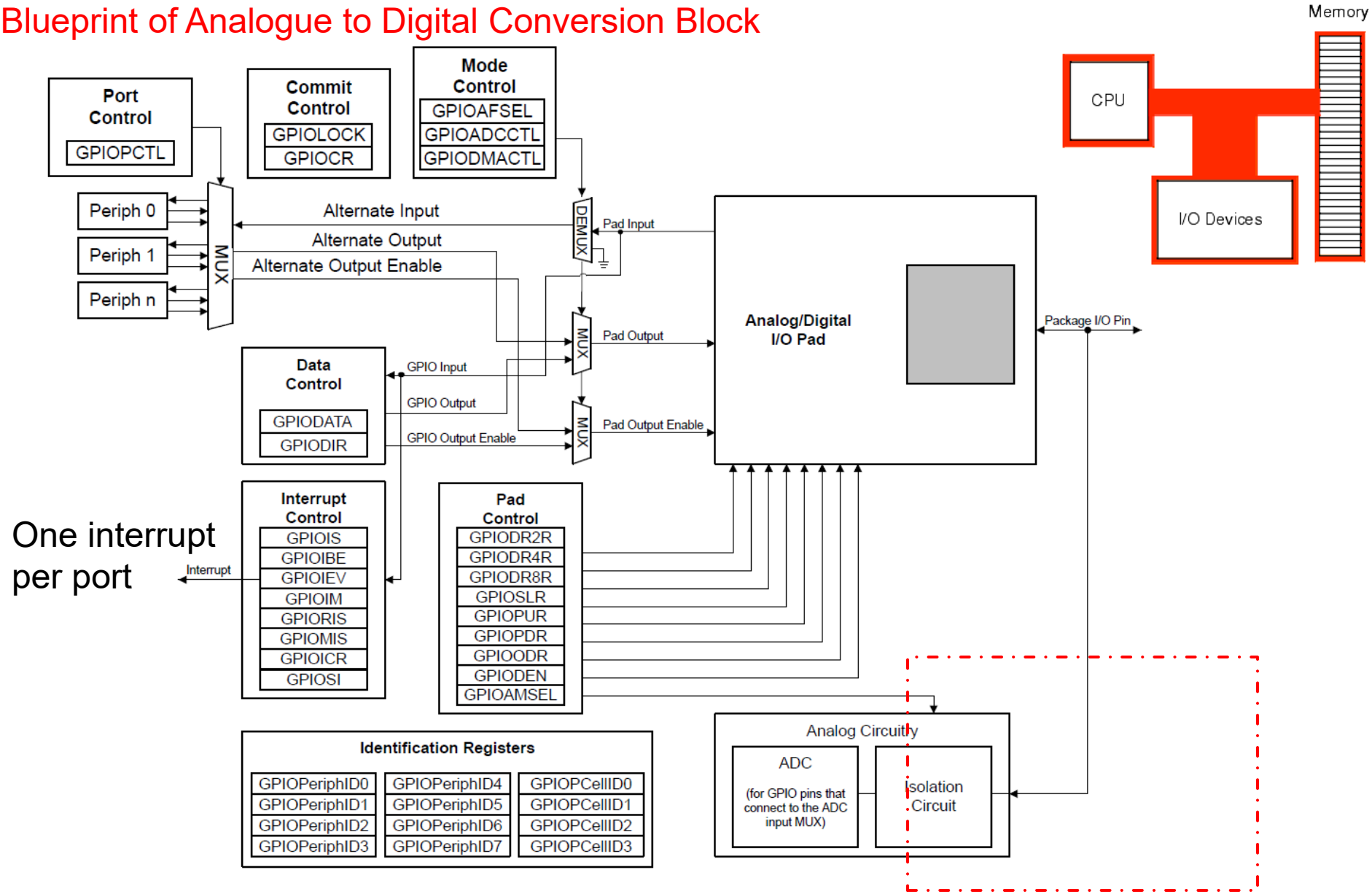
Blueprint of GPIO Pad



One interrupt per port

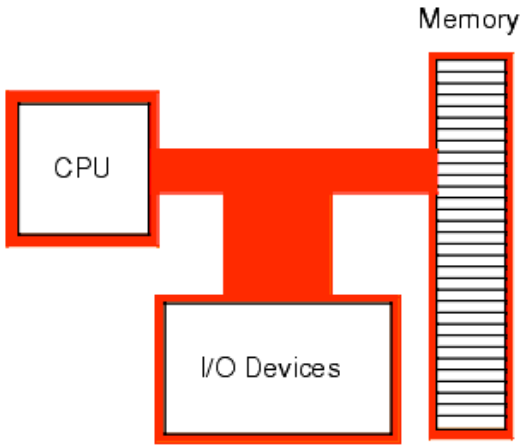
Each GPIO port is a separate hardware instantiation of the same physical block.

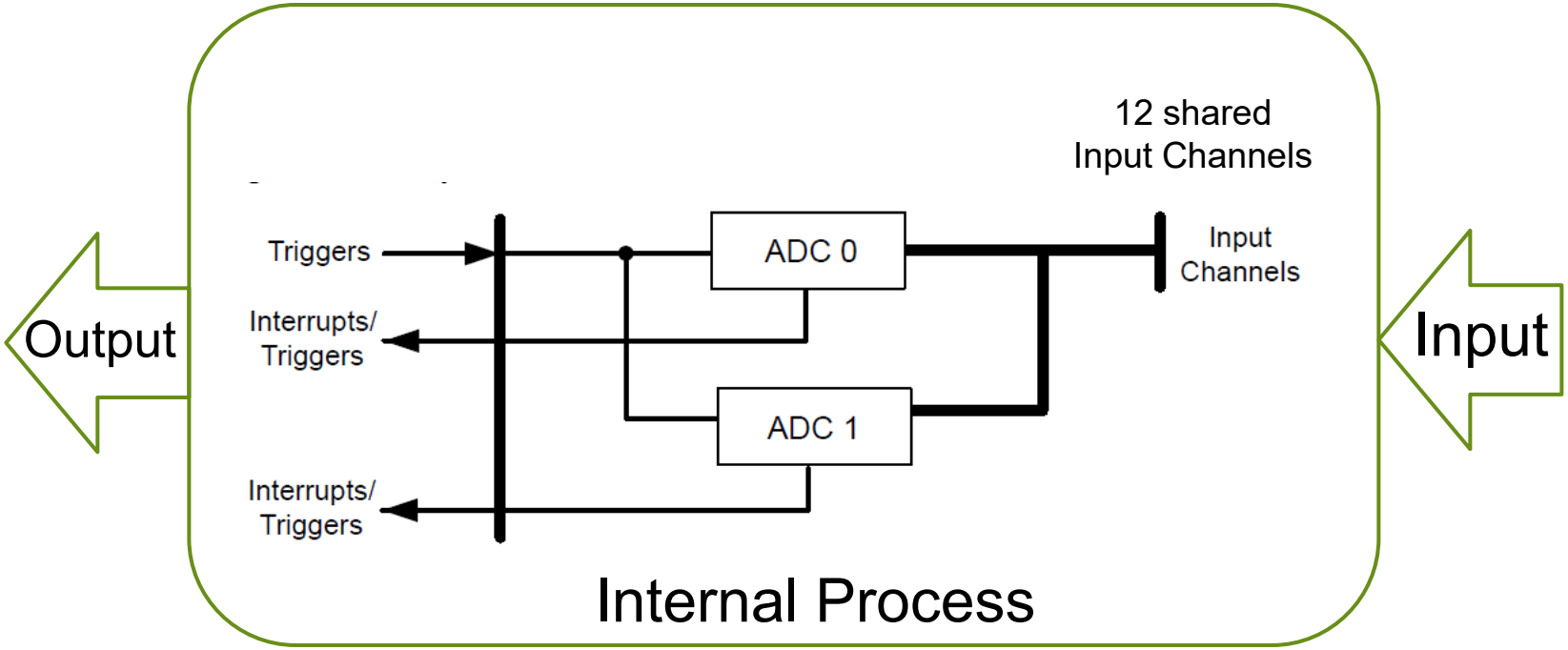
Blueprint of Analogue to Digital Conversion Block

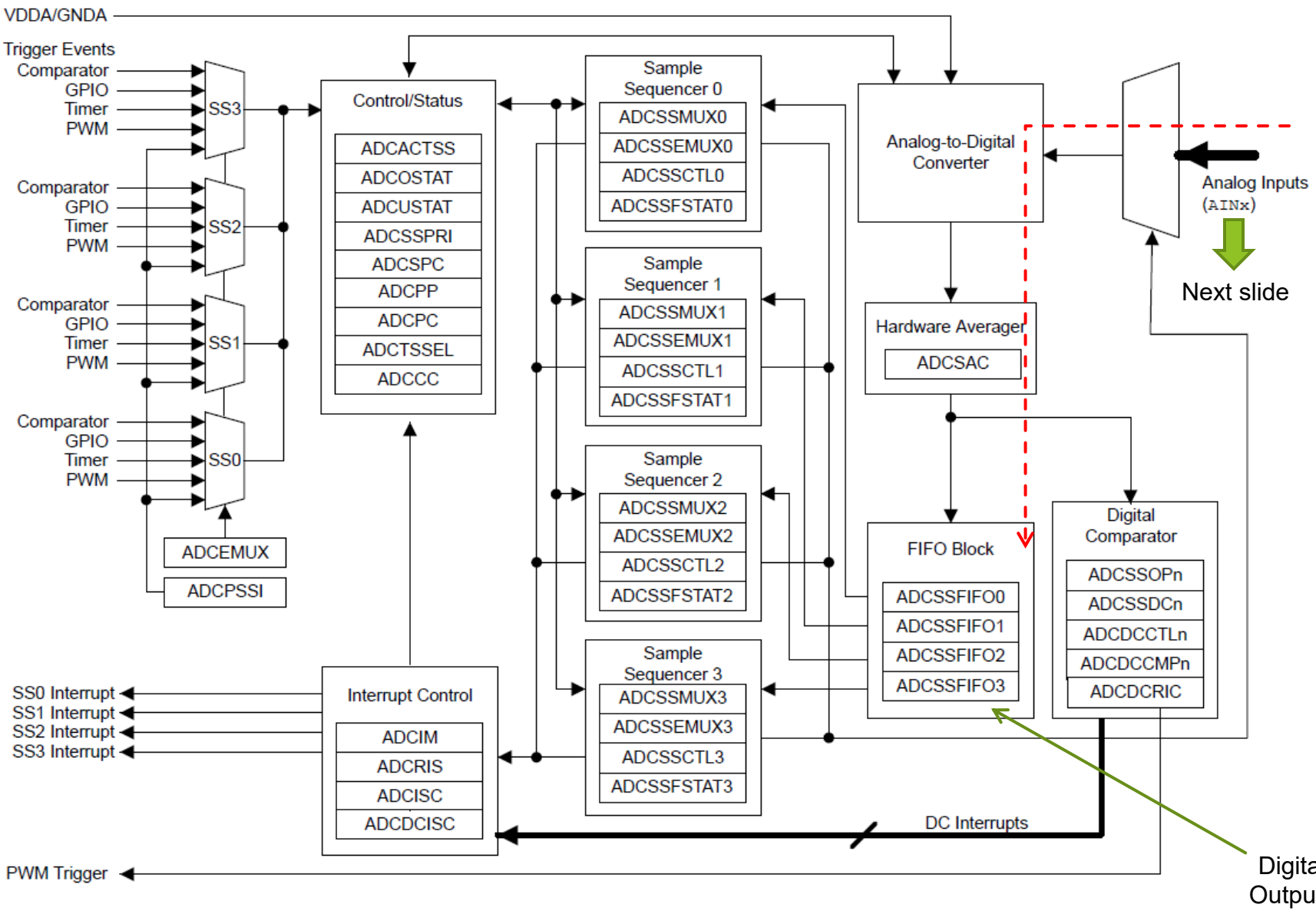


One interrupt per port

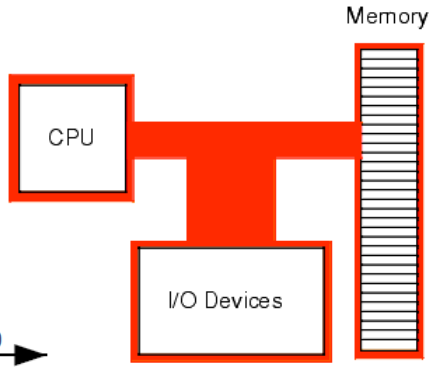
Blueprint of ARM Cortex M4's ADC





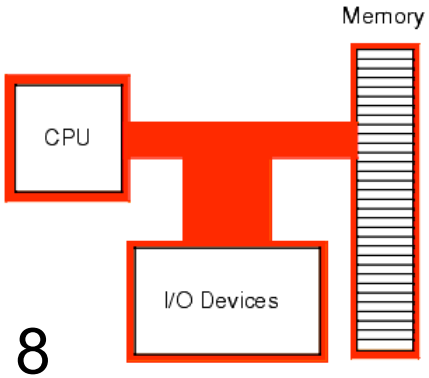

VDDA/GNDA
Trigger Events
Comparator
GPIO
Timer
PWM
SS3
SS2
SS1
SS0
ADCEMUX
ADCPSSI
Control/Status
ADCACTSS
ADCOSTAT
ADCUSTAT
ADCSSPRI
ADCSPC
ADCPP
ADCPC
ADCTSSEL
ADCCC
Sample Sequencer 0
ADCSSMUX0
ADCSSEMUX0
ADCSSCTL0
ADCSSFSTAT0
Sample Sequencer 1
ADCSSMUX1
ADCSSEMUX1
ADCSSCTL1
ADCSSFSTAT1
Sample Sequencer 2
ADCSSMUX2
ADCSSEMUX2
ADCSSCTL2
ADCSSFSTAT2
Sample Sequencer 3
ADCSSMUX3
ADCSSEMUX3
ADCSSCTL3
ADCSSFSTAT3
Analog-to-Digital Converter
Analog Inputs (AINx)
Next slide
Hardware Averager
ADCSAC
FIFO Block
ADCSSFIFO0
ADCSSFIFO1
ADCSSFIFO2
ADCSSFIFO3
Digital Comparator
ADCSSOPn
ADCSSDCn
ADCDCCTLn
ADCDCCMPn
ADCDCRIC
SS0 Interrupt
SS1 Interrupt
SS2 Interrupt
SS3 Interrupt
Interrupt Control
ADCIM
ADCRIS
ADCISC
ADCDCISC
DC Interrupts
PWM Trigger
Digital Outputs

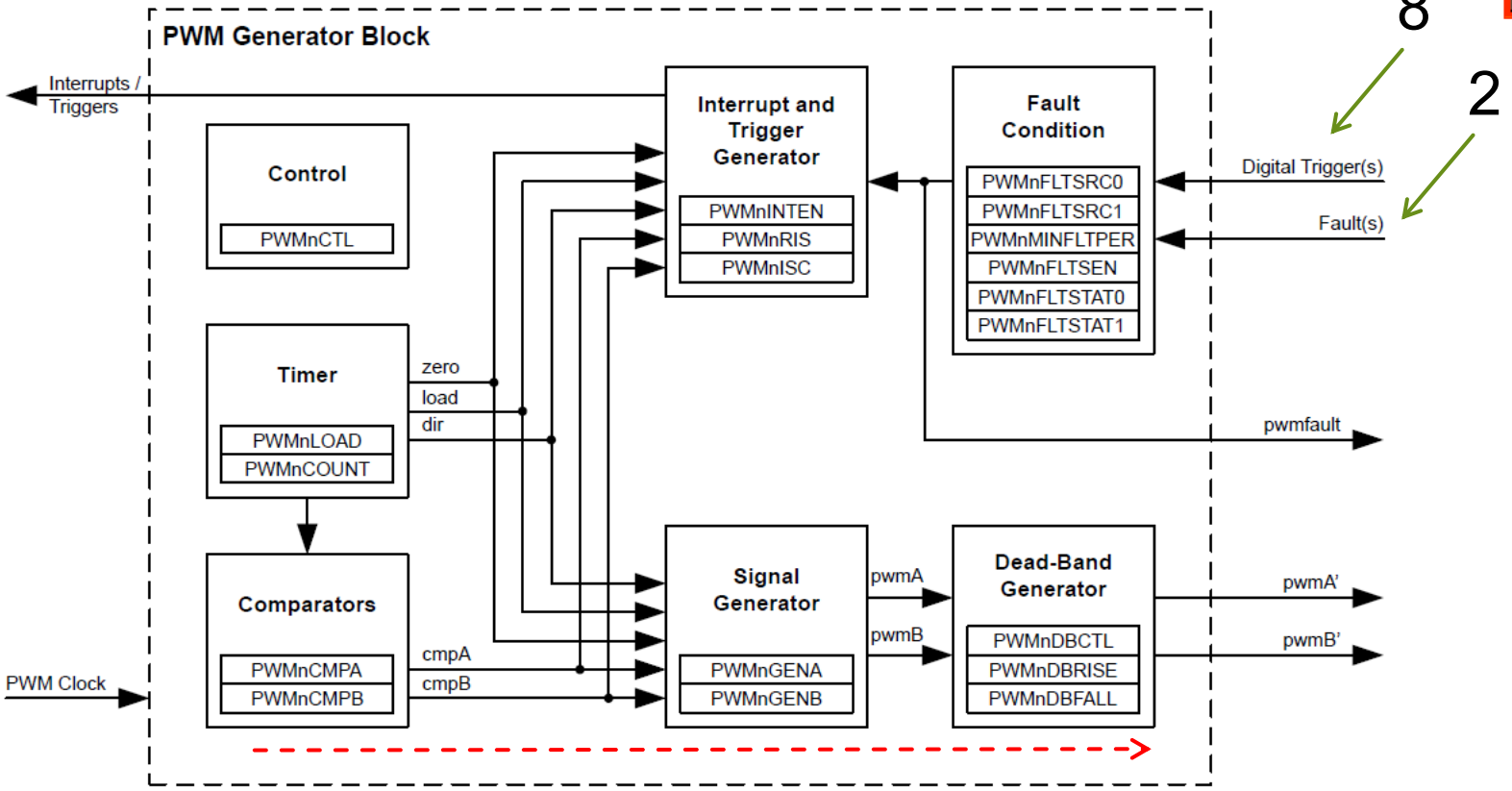
Blueprint of Cortex M4's PWM Module



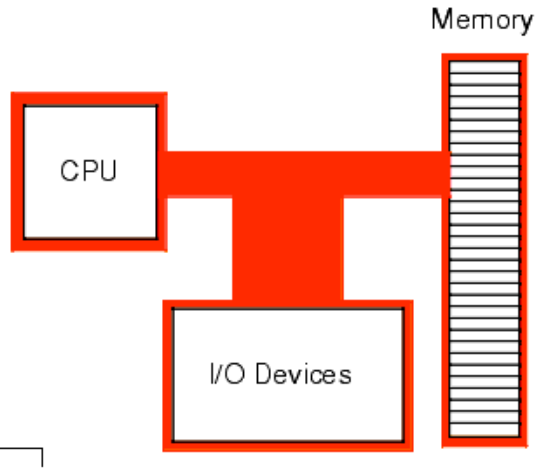
PWM Clock
Triggers / Faults
System Clock
Control and Status
PWMCTL
PWMSYNC
PWMSTATUS
PWMPP
Interrupt
PWMINTEN
PWMRIS
PWMISC
Interrupts
Triggers
Output
PWMENABLE
PWMINVERT
PWMFAULT
PWMFAULTVAL
PWMENUPD
PWM Generator 0
PWM Generator 1
PWM Generator 2
PWM Generator 3
pwm0A'
pwm0B'
pwm0fault
pwm1A'
pwm1B'
pwm1fault
pwm2A'
pwm2B'
pwm2fault
pwm3A'
pwm3B'
pwm3fault
PWM Output Control Logic
PWM 0
PWM 1
PWM 2
PWM 3
PWM 4
PWM 5
PWM 6
PWM 7

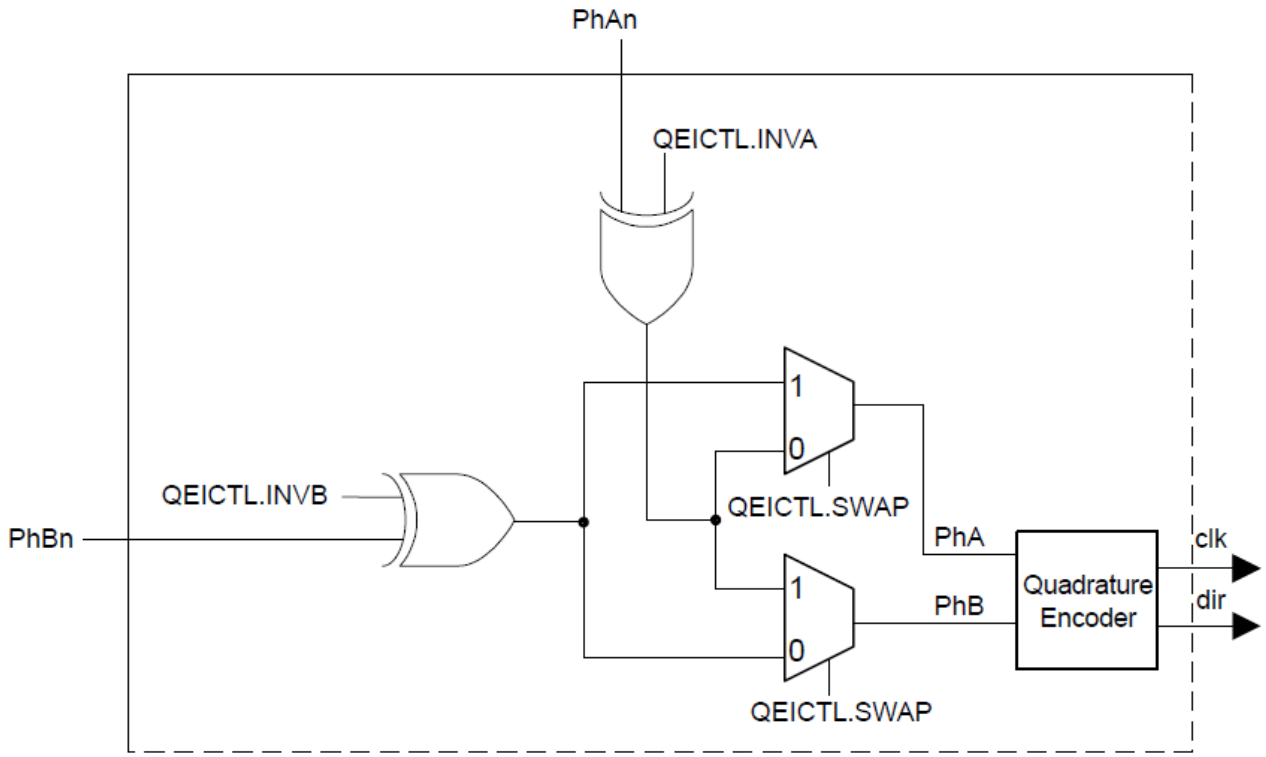
Blueprint of Each PWM Generator



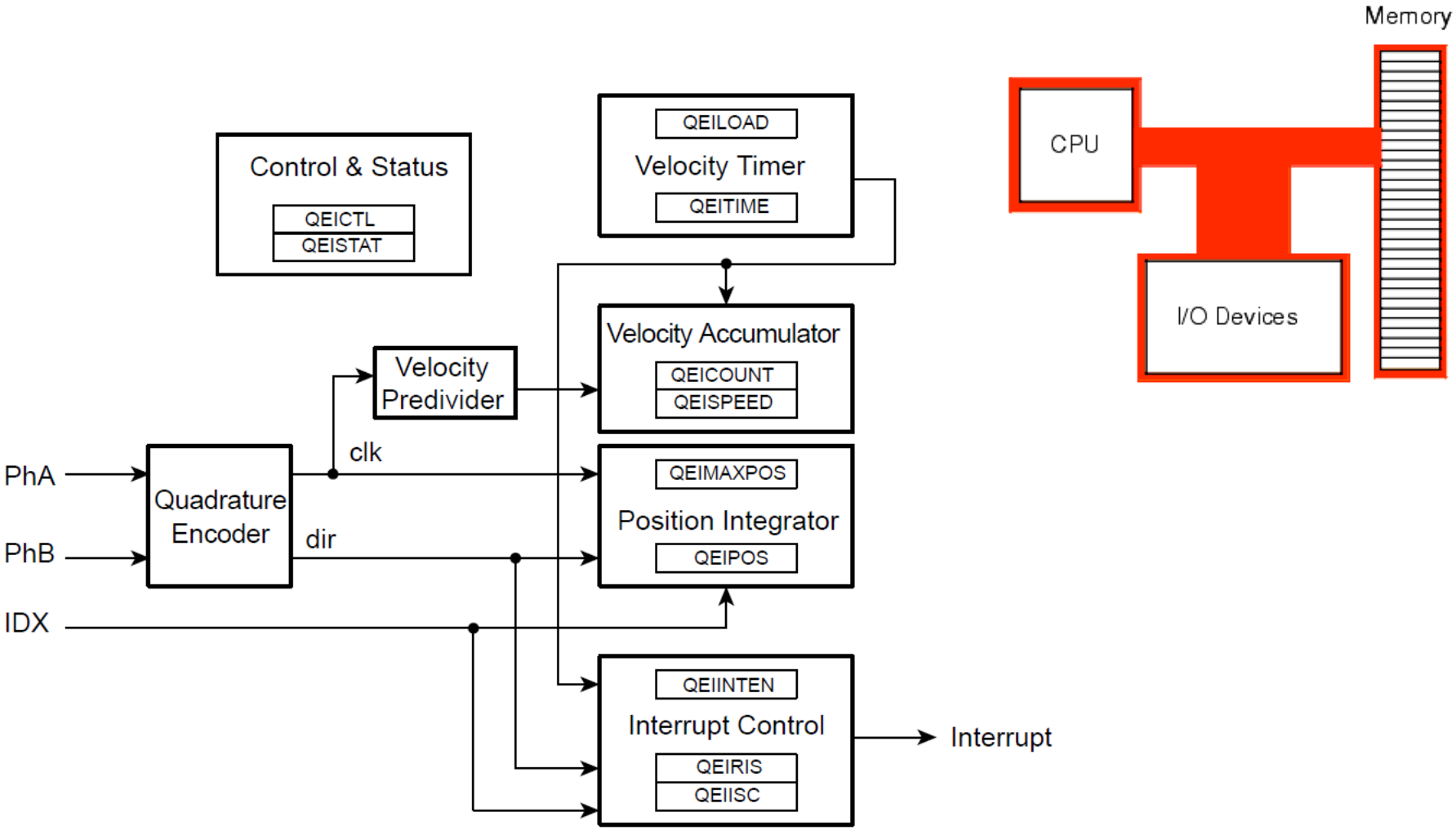


Blueprint of QEI Module

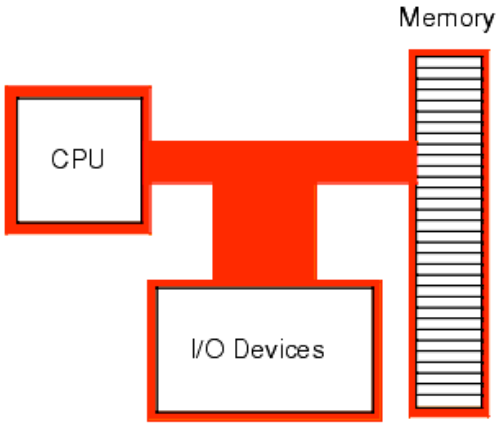


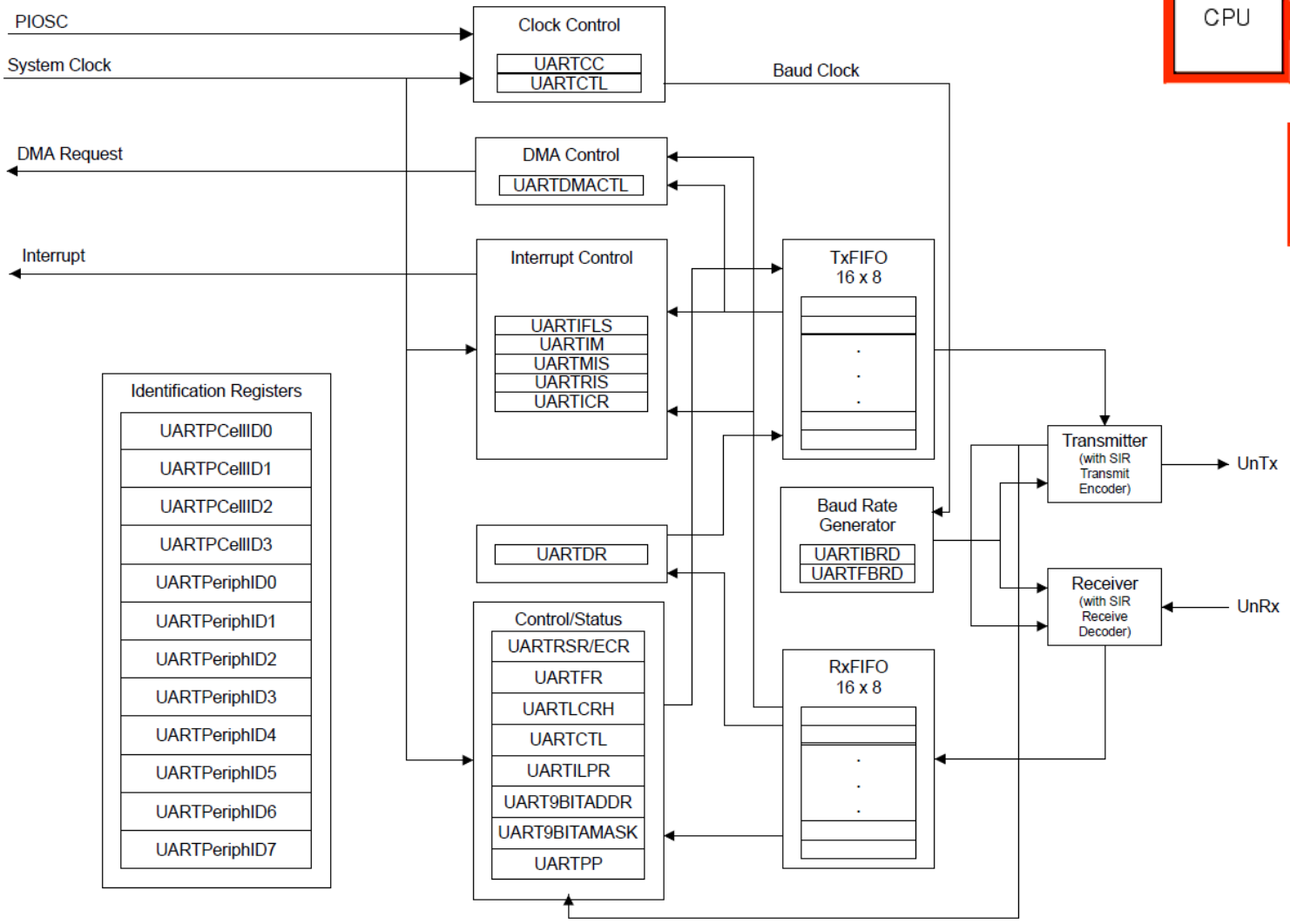


Blueprint of QEI Block Diagram

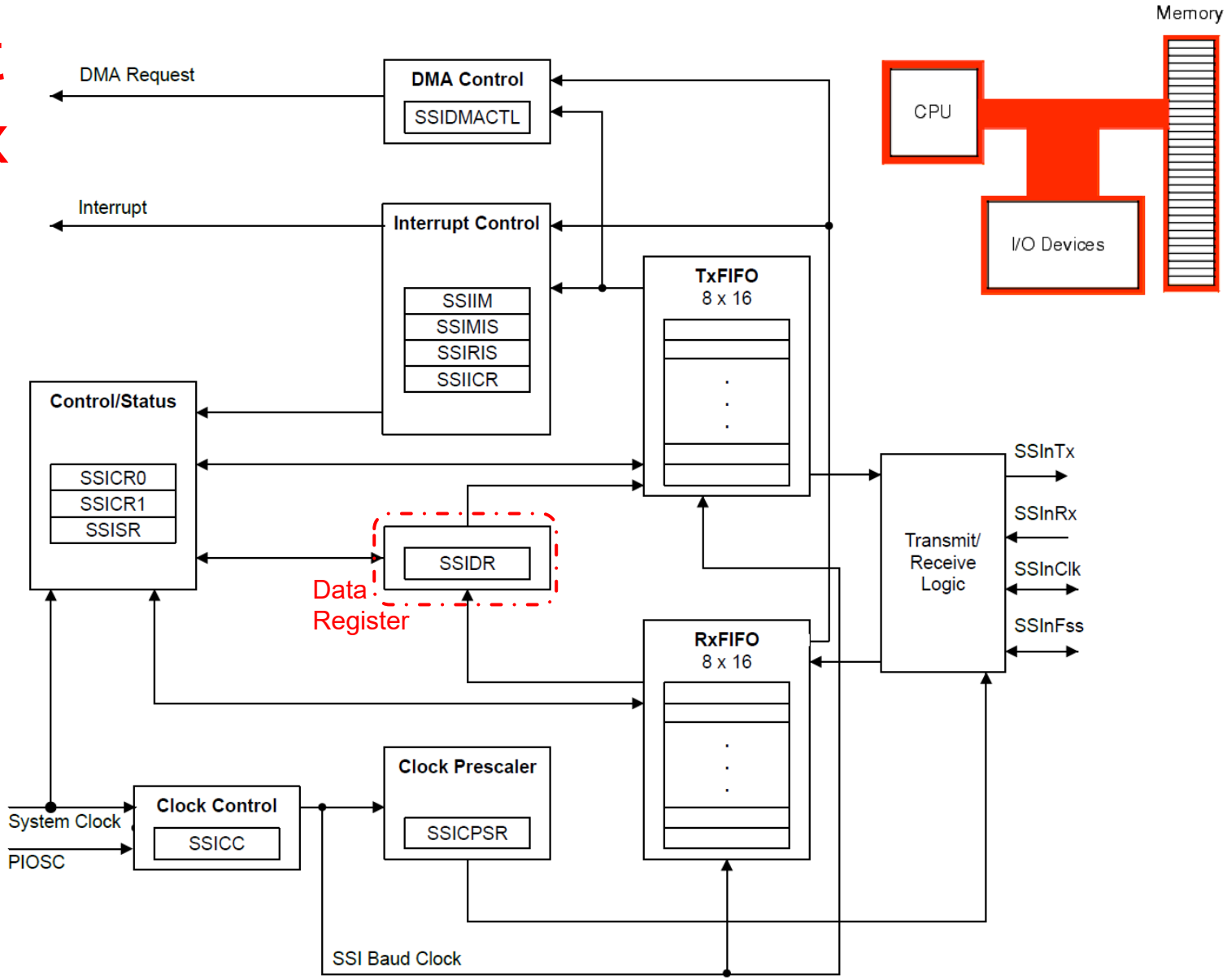


Blueprint of Cortex M4's UART

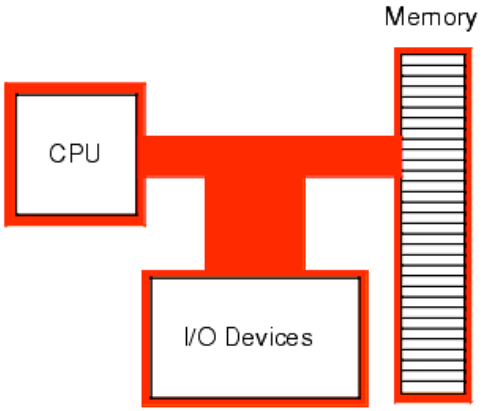


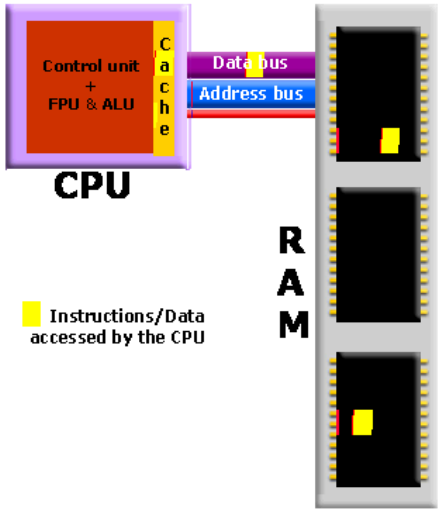


Blueprint of Cortex M4's SSI

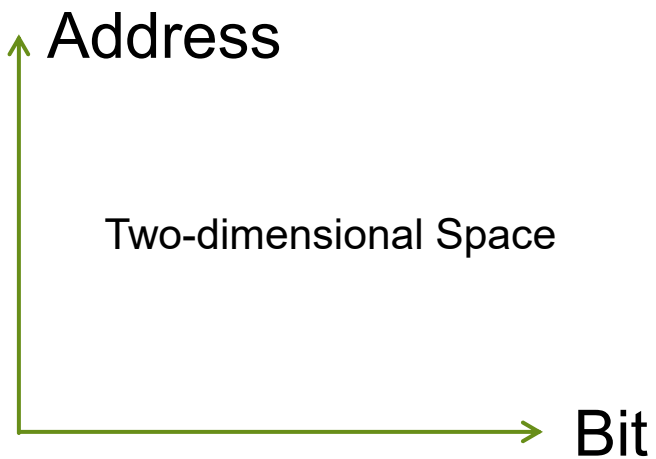


Summary





- Binary Logic Devices
- Memory Construction
- Memory Space in ARM
- Memory-Centric Operations
- Memory-Mapped I/O Devices



1. Memory Content
2. Memory Address
3. Memory Label

"Ask not what your country can do for you – ask what you can do for your country," - John F. Kennedy

"Do not think that you are needy – think that you are needed in the world", - Manis Friedman

"Study will make you knowledgeable, resourceful, and hence more needed", - Xie Ming

Thank You for Listening!