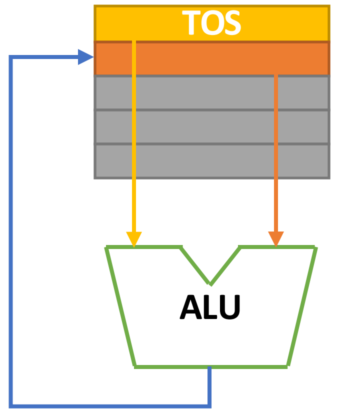
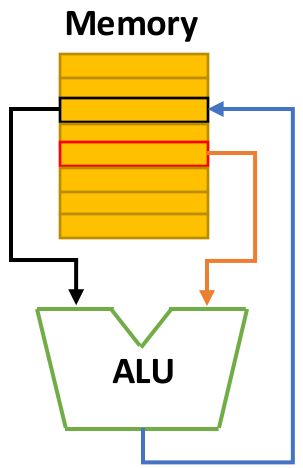
Memory model

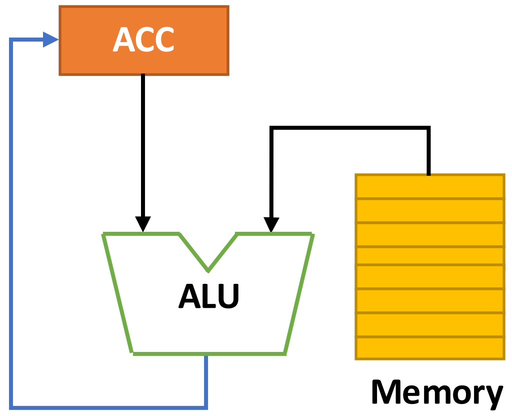
Memory model

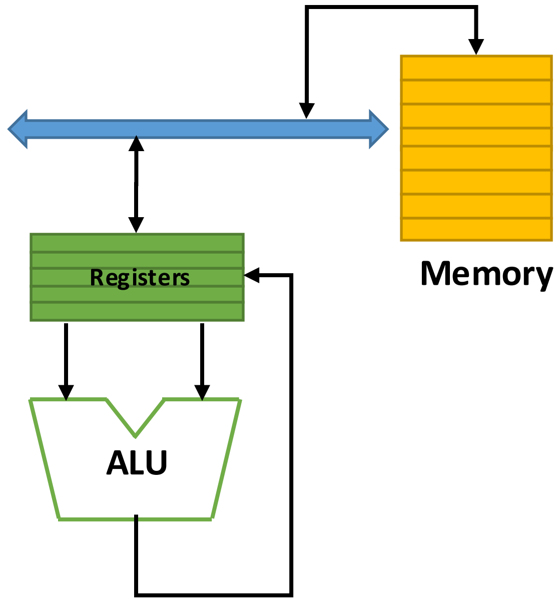
- Address space
  - 2^{32} 8bit bytes
  - 2^{31} 16bit half words
  - 2^{30} 32-bit words
- Architecture provides facilities for,
  - Generating exception on unaligned memory access
  - Restricting access by applications to specified areas of memory
  - Translating virtual address provided by executing instructions into physical address
  - Altering interpretation of word and half word data between big and little endian
  - Controlling order of access to memory
  - Controlling caches
  - Synchronizing access to shared memory by multiple processors

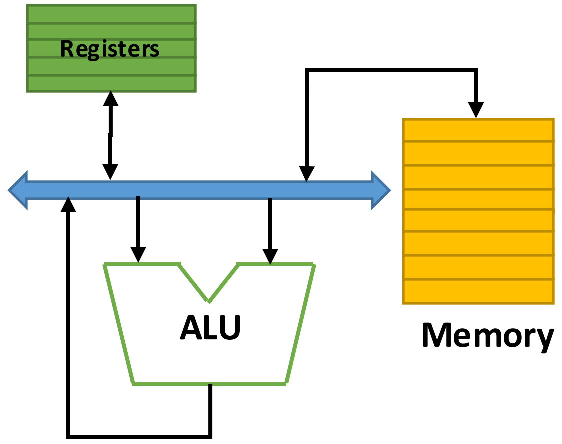
Stack Based Architecture







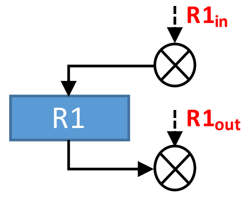




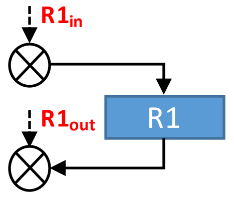
Register-Register based Architecture

Memory attributes

- Access Permissions
  - Read and Write
  - Read only (RO)
  - No Access
- Executable?
  - Can instructions be fetched from this location?
- Memory types
  - What kind of memory it is? (ROM, RAM, MMIO)
    - Normal
    - Device (always non-cacheable)
    - Strongly ordered
- Shareable → it does not provide any security features, its just used by hardware for cache coherency purpose
  - Is location is shared more than one *master*?
- Cacheable
  - Is location is cacheable or not?
- Security
  - Is location in secure memory?



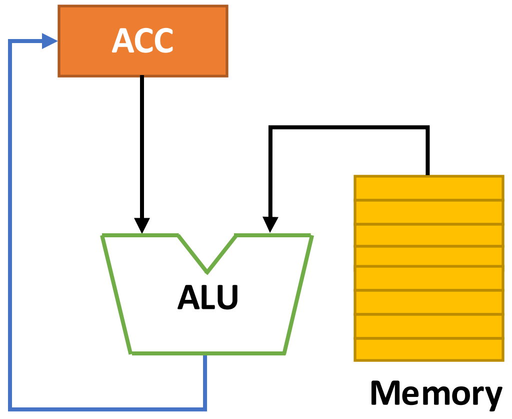
$R1_{in}$
R1
$R1_{out}$



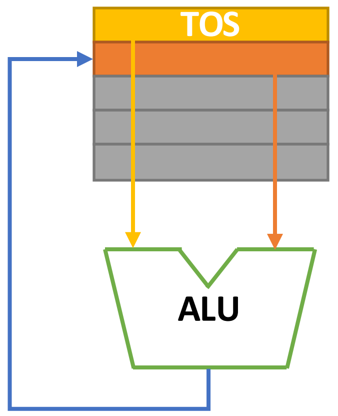
$R1_{in}$
R1
$R1_{out}$

Different Computer Architectures

Accumulator based Architecture

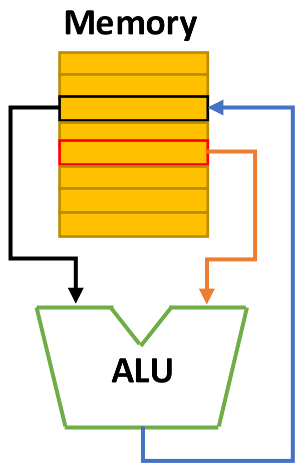


Stack based Architecture

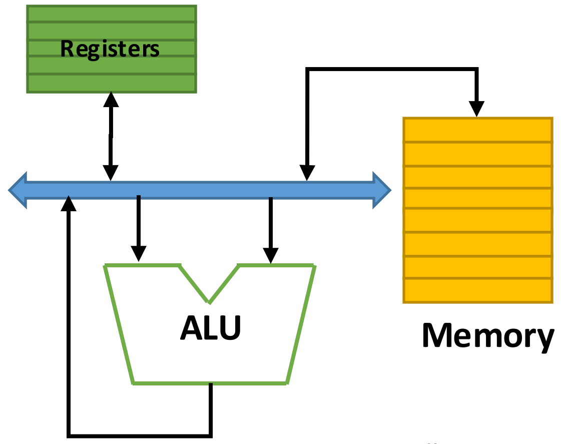


S.No	ISA type	Timeline	Customer
1	Accumulator based	1960's	EDSAC, IBM 1130
2	Stack based	1960-70	Burroughs 5000
3	Memory-Memory based	1970-80	IBM 360
4	Register-Memory based	1970-Present	Intel x86
5	Register-Register based	1960-Present	MIPS, CDC 6600, SPARC

Memory-Memory based Architecture

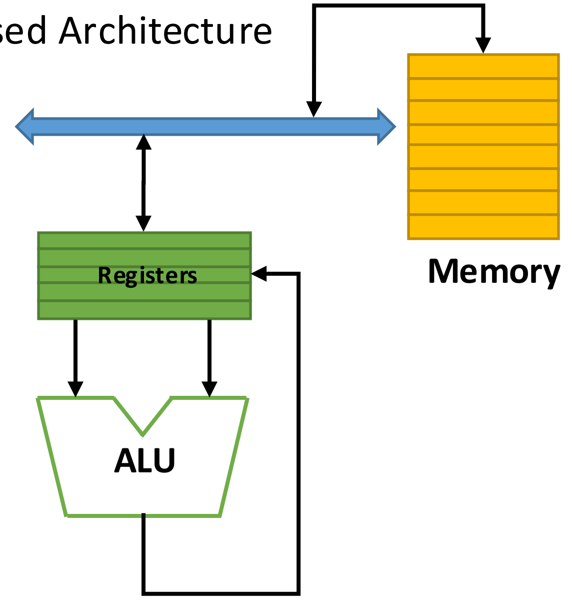


Register-Memory based Architecture



Parandhaman

Register-Register based Architecture

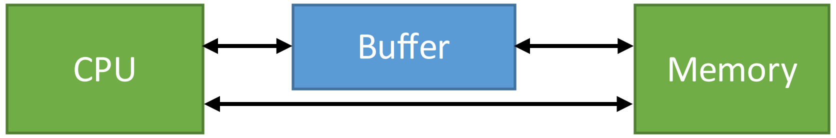


Memory types - Device

- Non-Gathering or Gathering (nG/G)
  - This property indicates whether combining several memory access into single memory access is allowed

Code	nG/G = 0 (Gathering Disabled)	nG/G = 1 (Gathering Enabled)
STR r0, [R4] STR r1, [R4] STR r2, [R4] STR r3, [R4]	Takes 4 memory cycles (note – 4 different bus cycles)	Single burst

- Non-reordering or Reordering (nR/R)
  - Out of order memory access is allowed or not?
- Early write acknowledgment (E)
  - Do we need to use write buffer or not? or write allocate or not?

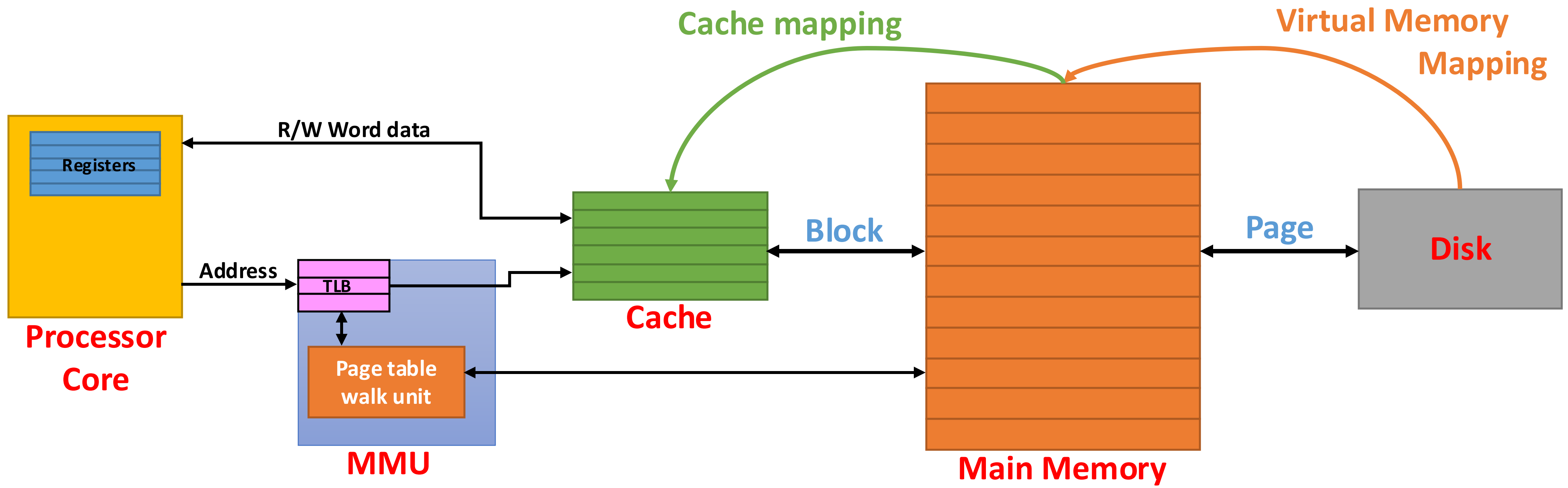


- Four device memory types
  - **nGnRnE** (~same as normal)
  - **nGnRE**
  - **nGRE**
  - **GRE**

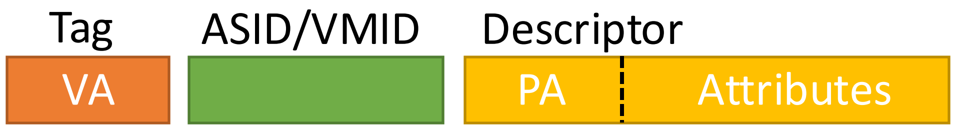
Memory types - Normal

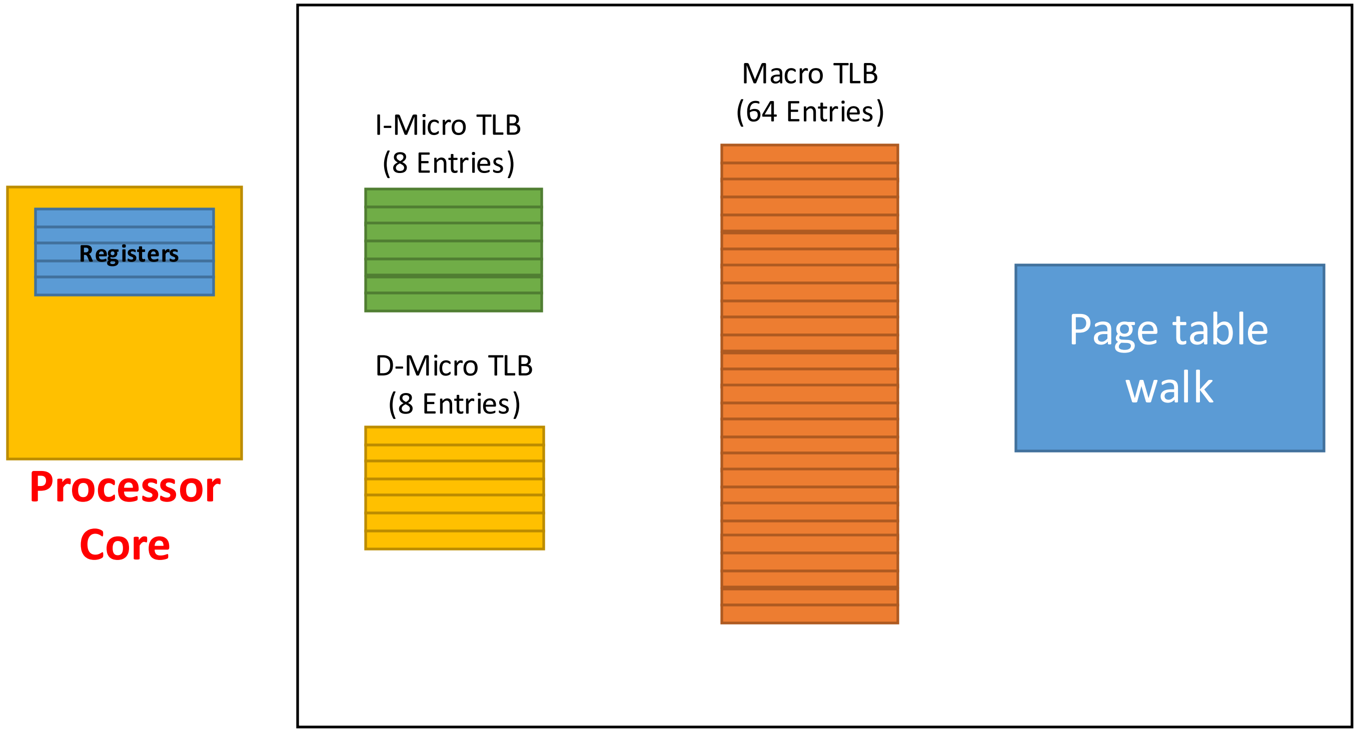
- Speculative access property
  - Hardware can read the data without any Load instructions
  - Basically hardware predicts this data may be needed in near future (by understanding previous access patterns)
  - It may use caches to store speculative memory access data
- Allocation property
  - When data needs to be cached, during read or during write or during both read and write?
    - Allocate on Read miss
    - Allocate on write miss
    - Allocate on both read or write miss
- Writing to cached locations
  - Write through
  - Write back
- Transience property
  - I think this only for code memory - **TBL**
  - Transience options
    - Transient – data only needed temporarily → will not use closest cache
    - Non-Transient– data is required for some time → will use closest cache
  - Based on Transient or non-transient processor will determine whether keeping the code in nearest cache is required or not?
    - Transient code will not use Level 1 Cache and it uses Level 2 cache only
    - Non-Transient code will use Level 1 Cache

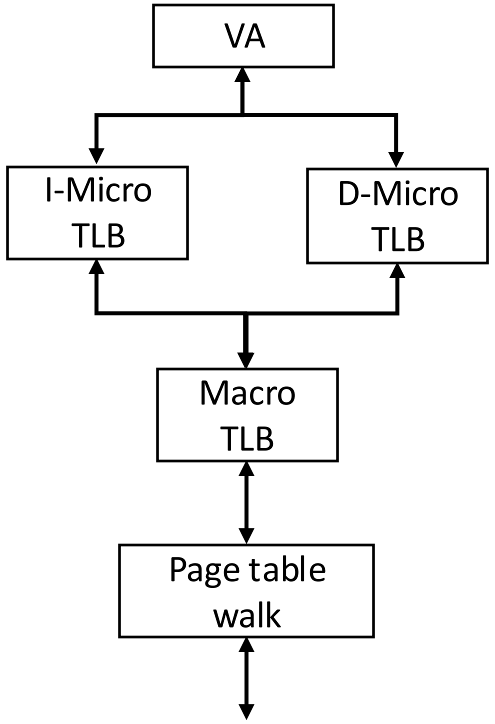
Read and Write operations from CPU to Memory

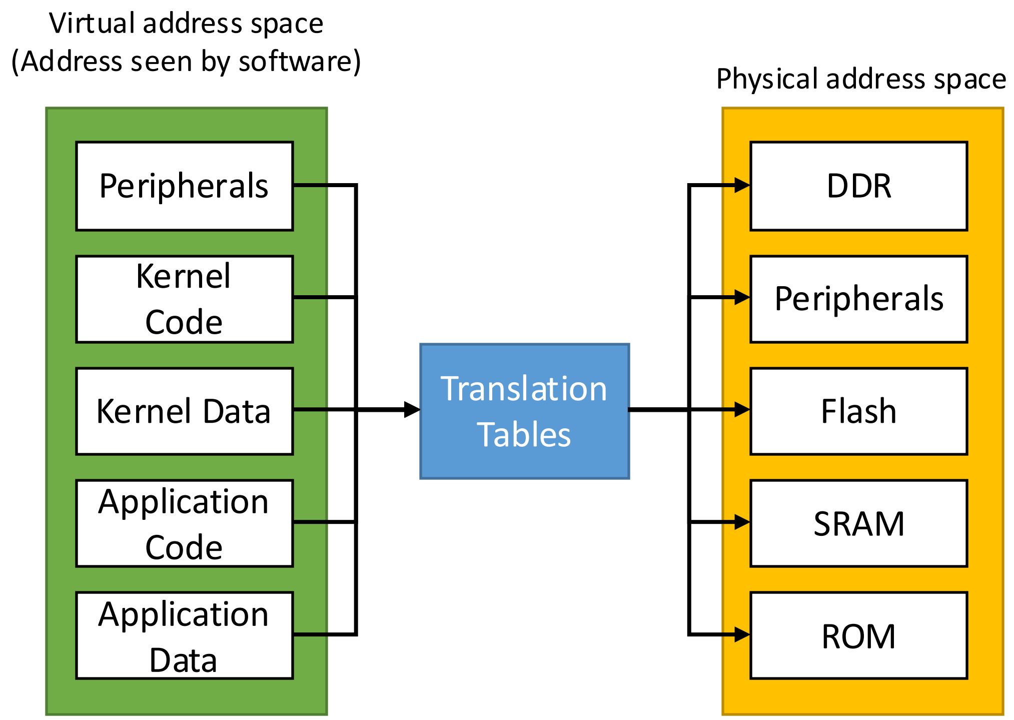


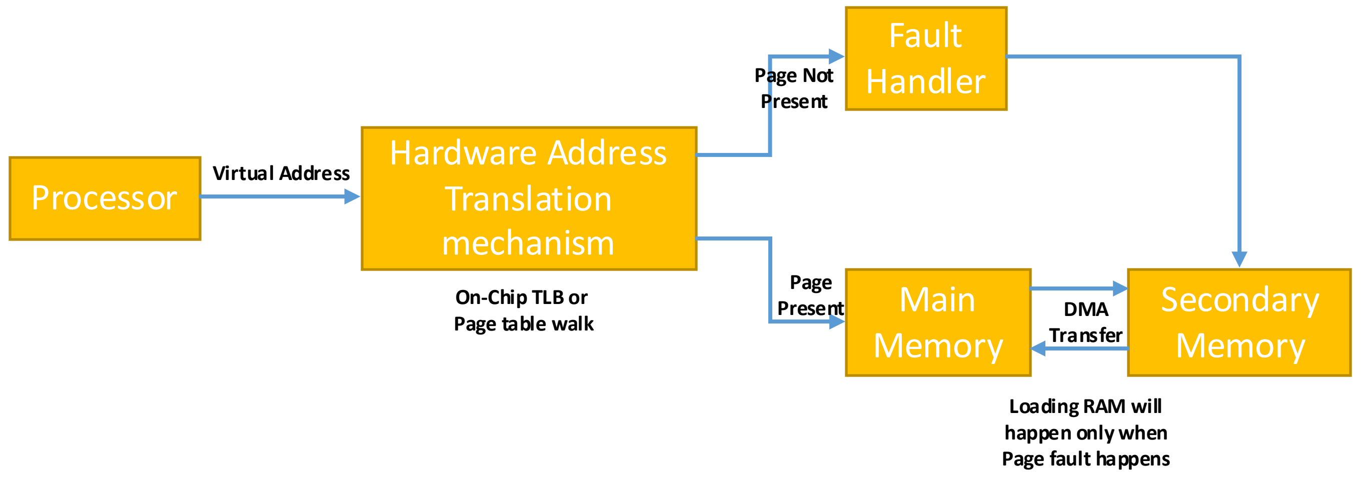
Memory data flow








Virtual address space
(Address seen by software)
Peripherals
Kernel Code
Kernel Data
Application Code
Application Data
Translation Tables
Physical address space
DDR
Peripherals
Flash
SRAM
ROM


Processor
Virtual Address
Hardware Address Translation mechanism
On-Chip TLB or Page table walk
Page Not Present
Fault Handler
Page Present
Main Memory
DMA Transfer
Secondary Memory
Loading RAM will happen only when Page fault happens

Virtual page number (12 bits)	Page offset (30 bits)

Page 0	PPA	nG(1)

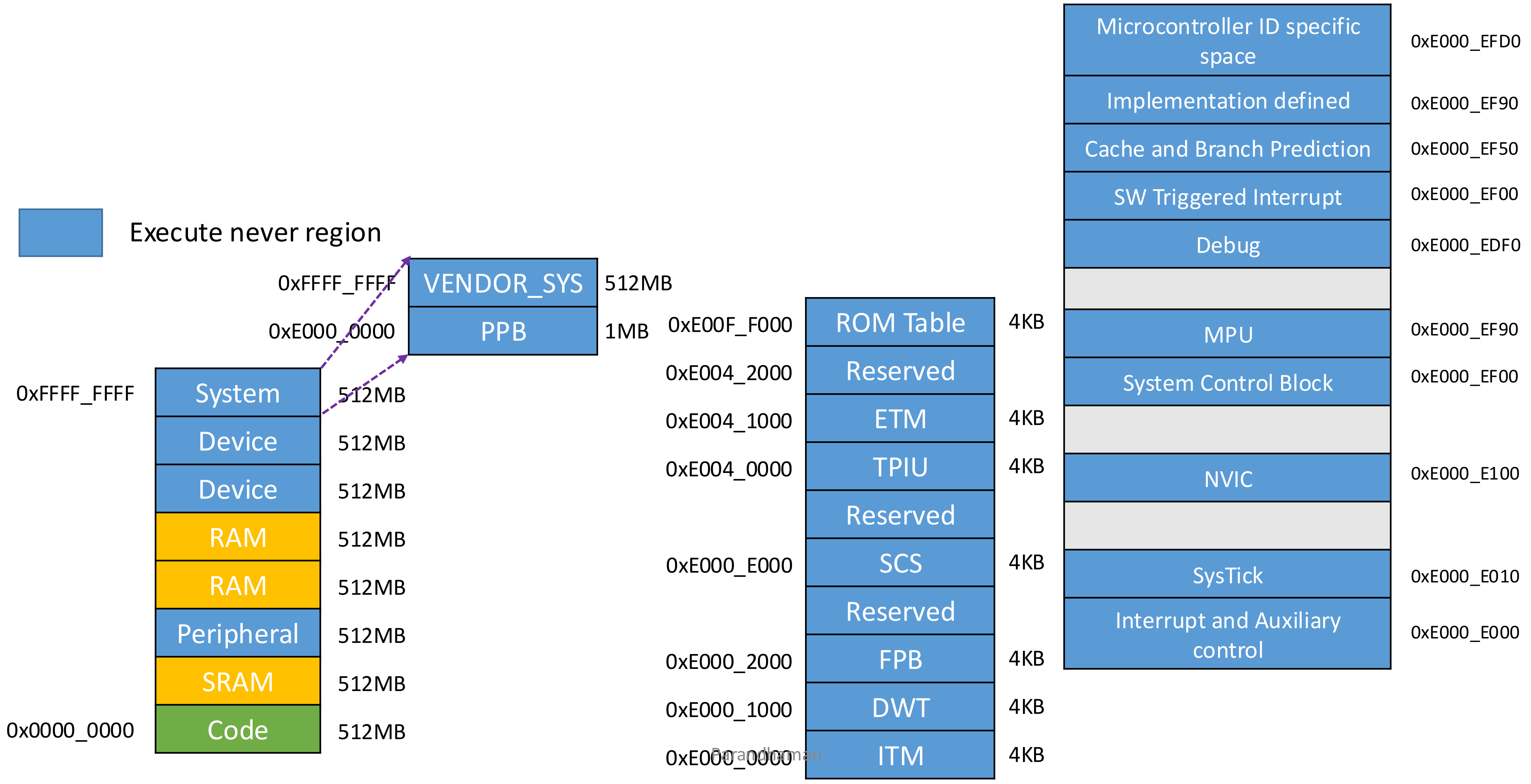
TTBR

Page 0	Physical page Address	Attributes
Page 1	PPA	
Page 2	PPA	
Page 3	PPA	
Page 4	PPA	
Page 5	PPA	
Page 6	PPA	
Page 7	PPA	
Page 8	PPA	
Page 9	PPA	
::::::	::::::	
::::::	::::::	
::::::	::::::	
Page n	PPA	

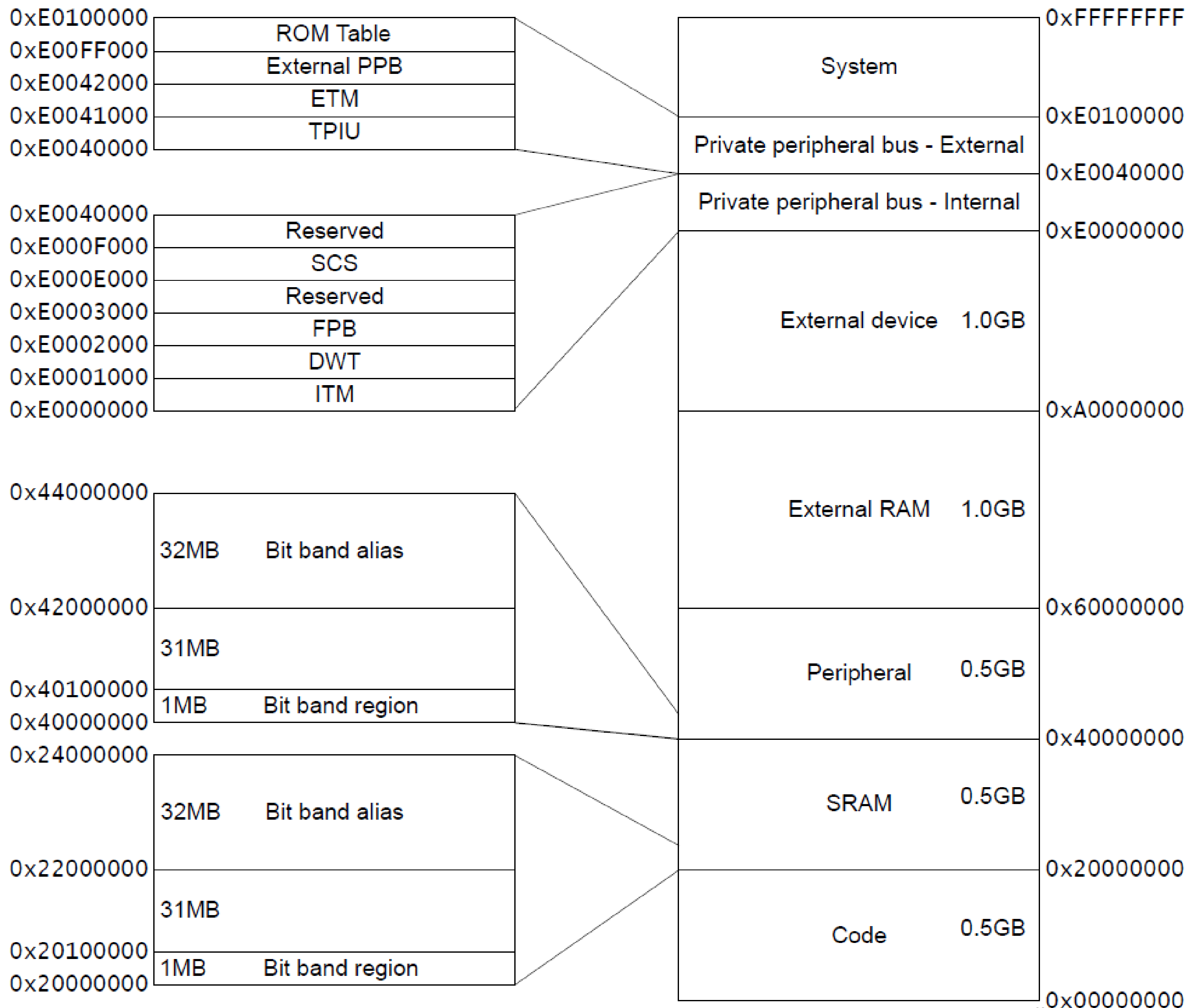
Parandhaman

Memory map

ARM v7M Memory Map



Memory types - Normal



Parandhaman

Code region was accessed through ICODE and DCOD

PPB Space was accessed through INTERNAL PPB