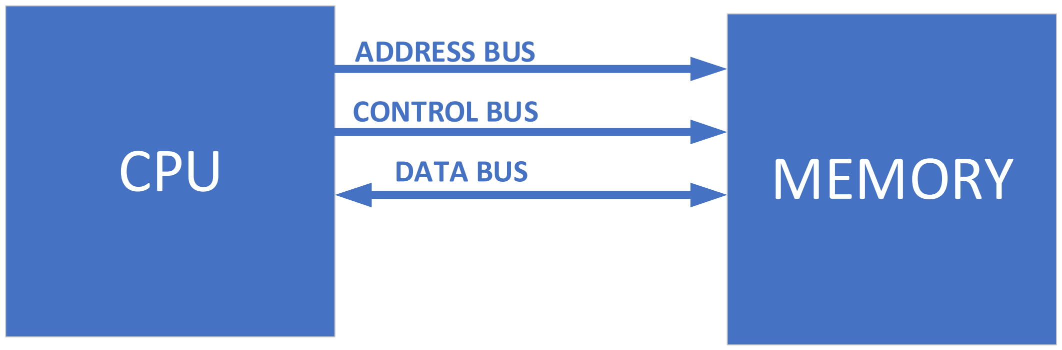
ACCESSING MEMORY

Dr. Russ Meier

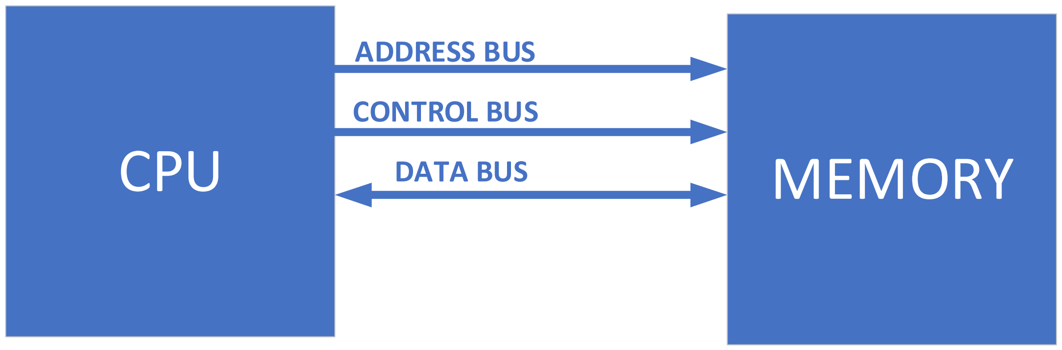
MEMORY BUSSES

- The CPU creates three numerical memory busses
  - An **address bus** requests access to memory locations
  - A **data bus** is used to move data between the CPU and the memory
  - A **control bus** contains signals controlling the memory chips

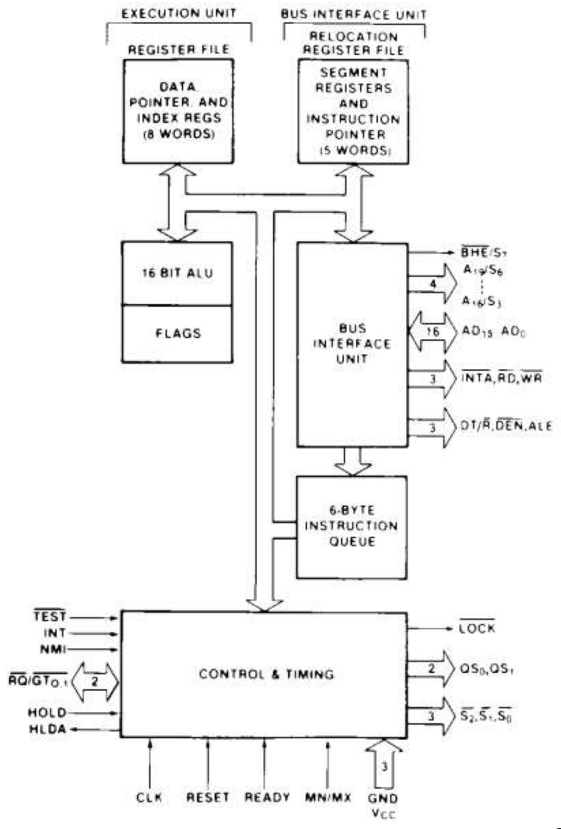


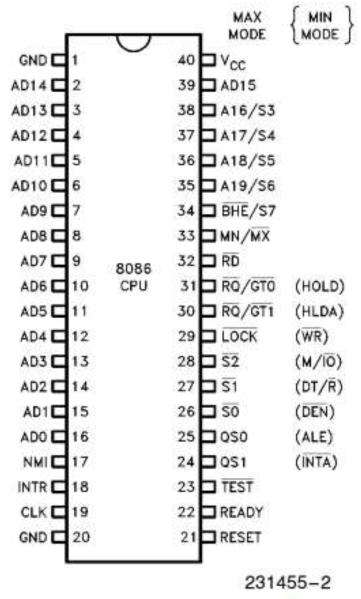
MEMORY BUSSES

- Control signals coordinate data movement using a **bus protocol**
  - Declare **address validity** so memories know the address bus contains a correct number
  - Declare **direction of transfer** using a read/write or a memory load signal
  - Control **speed of data flow** using clock signals or other flow control signals



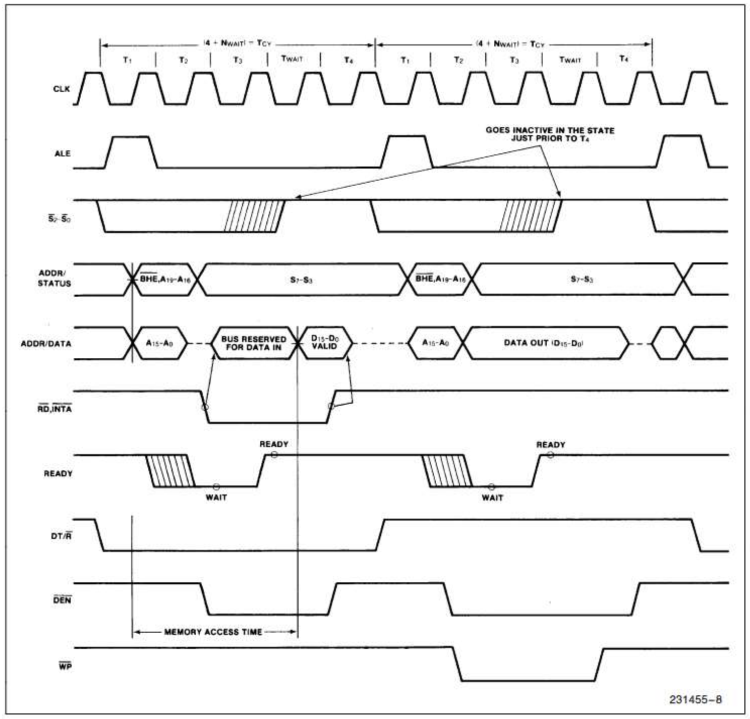
INTEL 8086 MEMORY BUS





40 Lead

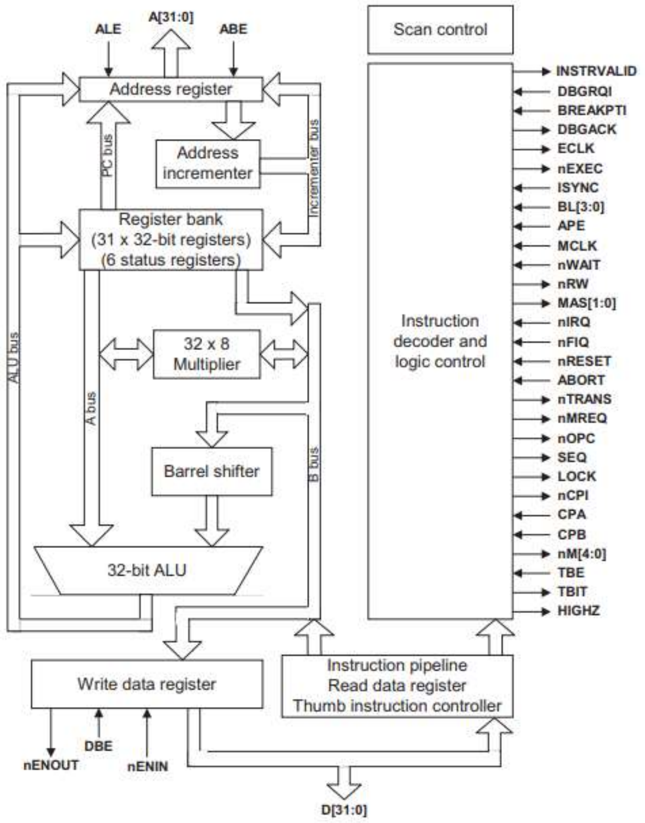
Figure 2. 8086 Pin Configuration

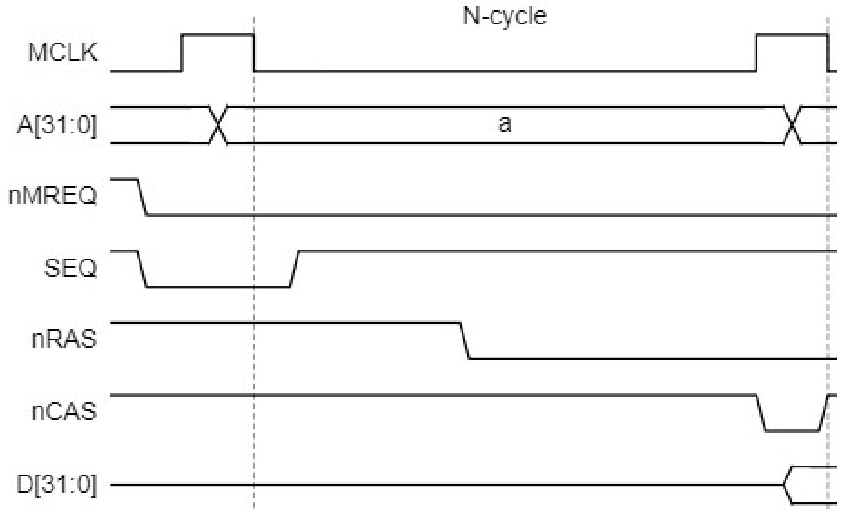


$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Characteristics
0 (LOW)	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1 (HIGH)	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (no bus cycle)

Source: Intel 8086 Datasheet, September 1990

ARM MEMORY BUS





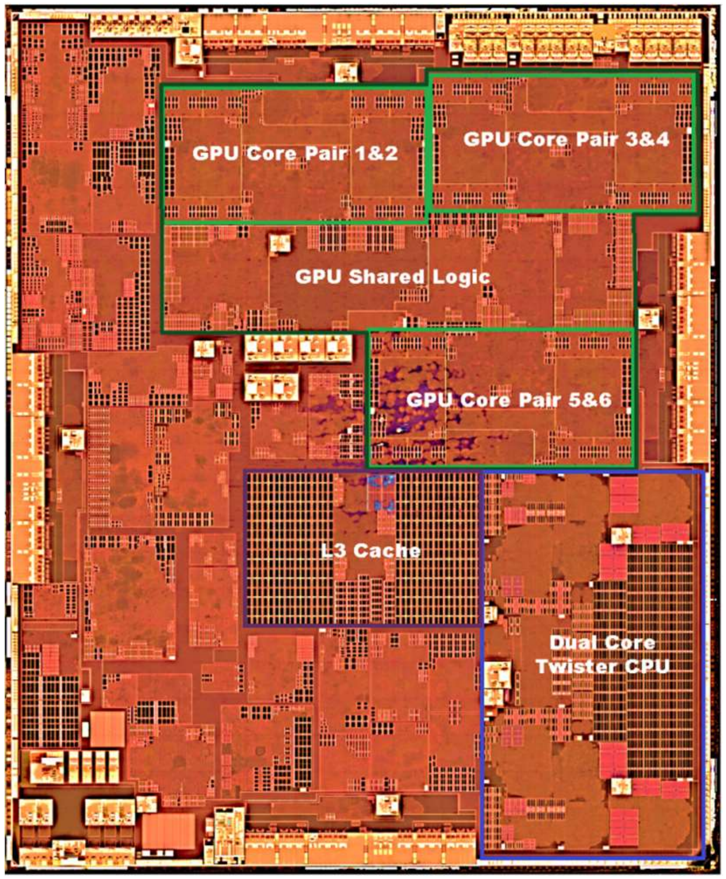
Bus cycle types

nMREQ	SEQ	Bus cycle type	Description
0	0	N-cycle	Nonsequential cycle
0	1	S-cycle	Sequential cycle
1	0	I-cycle	Internal cycle
1	1	C-cycle	Coprocessor register transfer cycle

Source: ARM7TDMI Technical Reference Manual, 2004

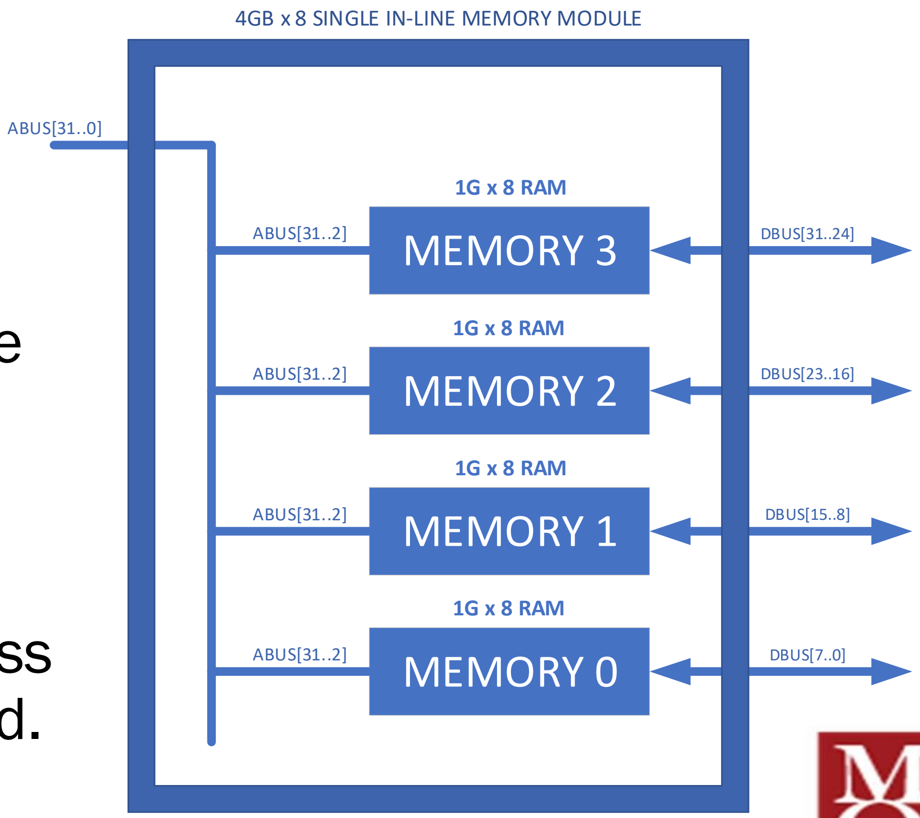
CACHE MEMORY

- Fabricated on CPU silicon die
- Multiple levels of cache exist
- This image is the Apple A9
- Dual-core ARMv8 CPU
- Level 1 and Level 2 in CPU Core
- L3 on-chip
- Accessed by the memory bus



MAIN MEMORY

- This example shows a full 32-bit 4GB address space using four 1GB memories.
- Each memory provides its byte at the specified address.
- Since four-bytes are provided, the memory is 32-bit aligned.
- Note that the upper 30 address bits specified the desired word.



MAIN MEMORY

- This example shows a DDR2 256MiB single in-line memory module (SIMM) for a personal computer.
- DDR2 uses a 64-bit data bus.
- Each chip is 32MiB of storage.
- Each chip provides its byte in the 64-bit word.
- 32MiB x 8 = 256MiB

LOAD AND STORE INSTRUCTIONS

- ARM accesses memory as a large array of 32-bit numbers.
- Consider the Java code:

j = A[0]

- The load-register instruction implements this equation as:

R[Rd] ← MEM[Rn,#0]

```
MOV R4,#0x80000000
LDR R0,[R4,#0]
```

REGISTER	VALUE
R0	0xAB94C600
R1	
R2	
R3	
R4	0x80000000

An array of five integers stored in memory

ADDRESS	DATA WORD
0x80000000	0xAB94C600
0x80000004	0xBADCAB02
0x80000008	0x01234567
0x8000000C	0x000FF99B
0x80000010	0xFEDCBA98

R4 is a memory address
R4 is a “pointer” into memory
R4 is a “reference” to the memory location

LOAD AND STORE INSTRUCTIONS

- ARM allows an index of 0 to be omitted in assembly language.
- Consider the Java code:

  j = A[0]

- The load-register instruction implements this equation as:

  R[Rd] ← MEM[Rn]

```
MOV R4,#0x80000000
LDR R0,[R4]
```

REGISTER	VALUE
R0	0xAB94C600
R1	
R2	
R3	
R4	0x80000000

An array of five integers stored in memory

ADDRESS	DATA WORD
0x80000000	0xAB94C600
0x80000004	0xBADCAB02
0x80000008	0x01234567
0x8000000C	0x000FF99B
0x80000010	0xFEDCBA98

R4 is a memory address
R4 is a “pointer” into memory
R4 is a “reference” to the memory location

LOAD AND STORE INSTRUCTIONS

- Consider the Java code:

  j = A[3]

- The load-register instruction implements this equation as:

  R[Rd] ← MEM[Rn,#12]

```
MOV R4,#0x80000000
LDR R0,[R4,#12]
```

REGISTER	VALUE
R0	0x000FF99B
R1	
R2	
R3	
R4	0x80000000

An array of five integers stored in memory

ADDRESS	DATA WORD
0x80000000	0xAB94C600
0x80000004	0xBADCAB02
0x80000008	0x01234567
0x8000000C	0x000FF99B
0x80000010	0xFEDCBA98

R4 is a memory address
R4 is a “pointer” into memory
R4 is a “reference” to the memory location

LOAD AND STORE INSTRUCTIONS

- In this example, R4 is the pointer.
- R4 can also be called the base address.
- The displacement is 12.

```
MOV R4,#0x80000000
LDR R0,[R4,#12]
```

- The final memory address is calculated as base address + displacement. The final memory address is called the *effective memory address.*

An array of five integers stored in memory

ADDRESS	DATA WORD
0x80000000	0xAB94C600
0x80000004	0xBADCAB02
0x80000008	0x01234567
0x8000000C	0x000FF99B
0x80000010	0xFEDCBA98

R4 is a memory address
R4 is a “pointer” into memory
R4 is a “reference” to the memory location

ADDRESSING MODE

- We have now seen three addressing modes in class.

ADDRESSING MODE	EXAMPLE	ADDRESS IN MEMORY	PSEUDO-CODE
Literal	MOV R4,#10	None needed	INT J = 10
Base	LDR R4,[R2]	R2 + 0	INT J = A[0]
Base with displacement	LDR R4,[R2,#16]	R2 + 16	INT J = A[4]

Base with displacement is also called base with offset

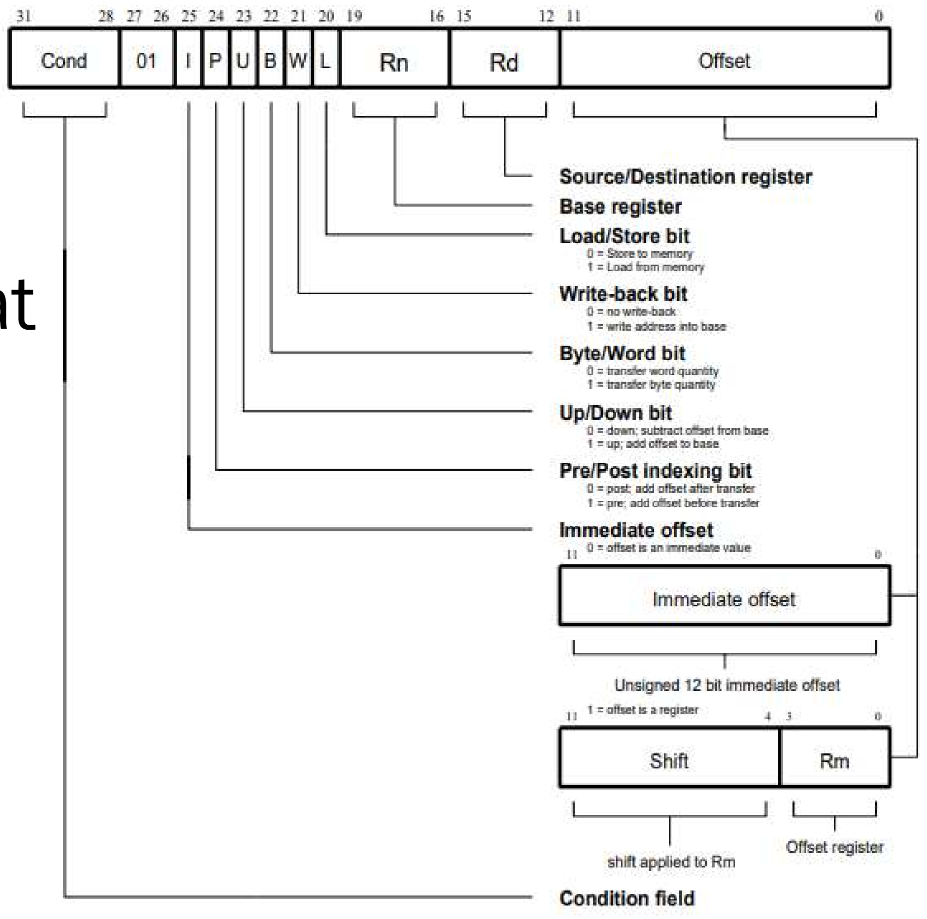
ENCODING LDR AND STR INSTRUCTIONS

EXAMPLE	ADDRESSING MODE	INDEX MODE	MEMORY ADDRESS	OP	$\bar{I}$	P	U	B	W	L
LDR R4,[R5,R6]	BASE + OFFSET	PRE-INDEX ADD R6	R5 + R6	01	1	1	1	0	0	1
LDR R4,[R5]	BASE + OFFSET	PRE-INDEX ADD #0	R5	01	0	1	1	0	0	1
LDR R4,[R5,#12]	BASE + OFFSET	PRE-INDEX ADD #12	R5 + 12	01	0	1	1	0	0	1
STR R4,[R5,#-20]	BASE + OFFSET	PRE-INDEX ADD #-20	R5 – 20	01	0	1	0	0	0	0

These are the only types of LDR and STR used in CE1921

ENCODING LDR AND STR INSTRUCTIONS

ARM Data Sheet Image Showing Encoding Format



31 28 27 26 25 24 23 22 21 20 19 16 15 12 11 0
Cond
01
I
P
U
B
W
L
Rn
Rd
Offset
Source/Destination register
Base register
Load/Store bit
0 = Store to memory
1 = Load from memory
Write-back bit
0 = no write-back
1 = write address into base
Byte/Word bit
0 = transfer word quantity
1 = transfer byte quantity
Up/Down bit
0 = down; subtract offset from base
1 = up; add offset to base
Pre/Post indexing bit
0 = post; add offset after transfer
1 = pre; add offset before transfer
Immediate offset
0 = offset is an immediate value
11 0
Immediate offset
Unsigned 12 bit immediate offset
1 = offset is a register
11 4 3 0
Shift
Rm
shift applied to Rm
Offset register
Condition field