Fundamentals of Computer Architecture

A Review Of Chapters 1 to 7

OVERVIEW

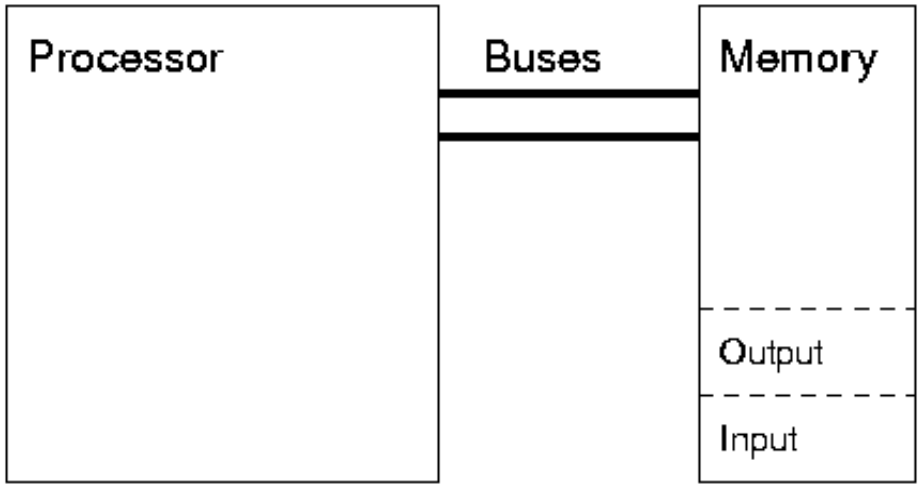
- This presentation includes:
  - Introducing The Processor
  - Fundamental Concepts I - Data Representation
  - Fundamental Concepts II - Digital Electronic Circuits
  - Registers
  - The ALU
  - Buses
  - Memory

Introducing The Processor

- Computer are everywhere
- Definition:
  - It must take *input* of some sort;
  - It must produce *output* of some sort;
  - It must *process* the information somehow;
  - It must have some sort of *information store*;
  - It must have some way of *controlling* what it does.
- Most computers are embedded in other devices

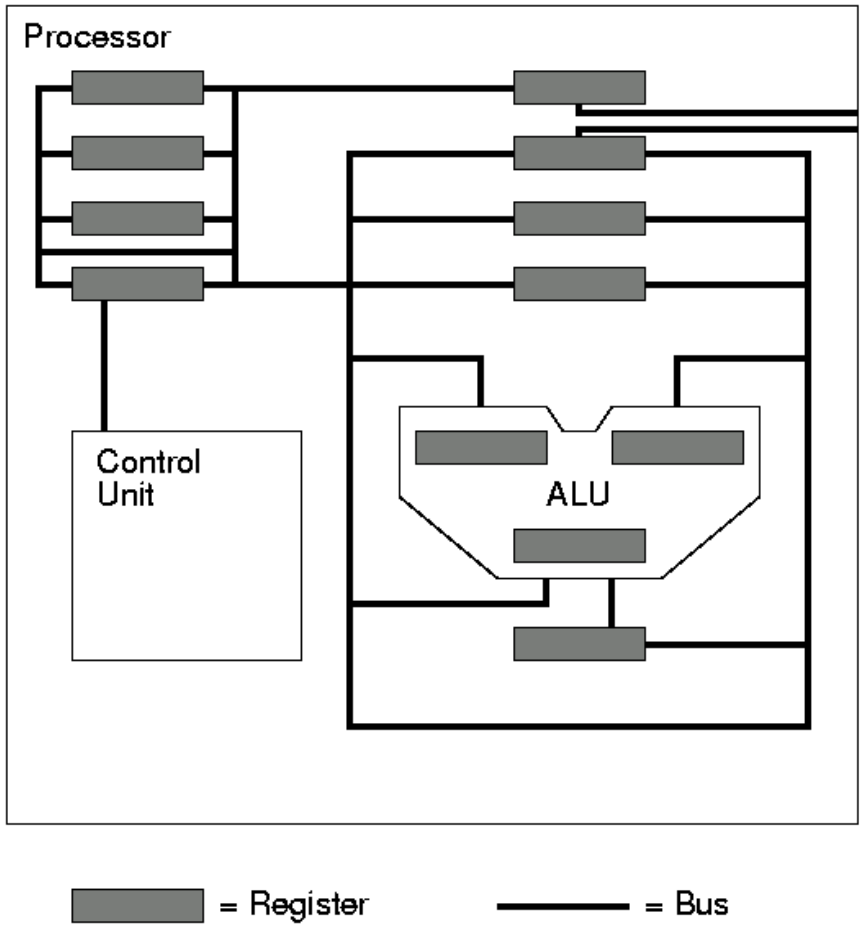
Introducing The Processor

- A *von Neumann architecture*,
- We need:
  - A processor - to process information, and to control the system;
  - Memory - for data and instruction storage;
  - Some form of input device; Some form of output device.

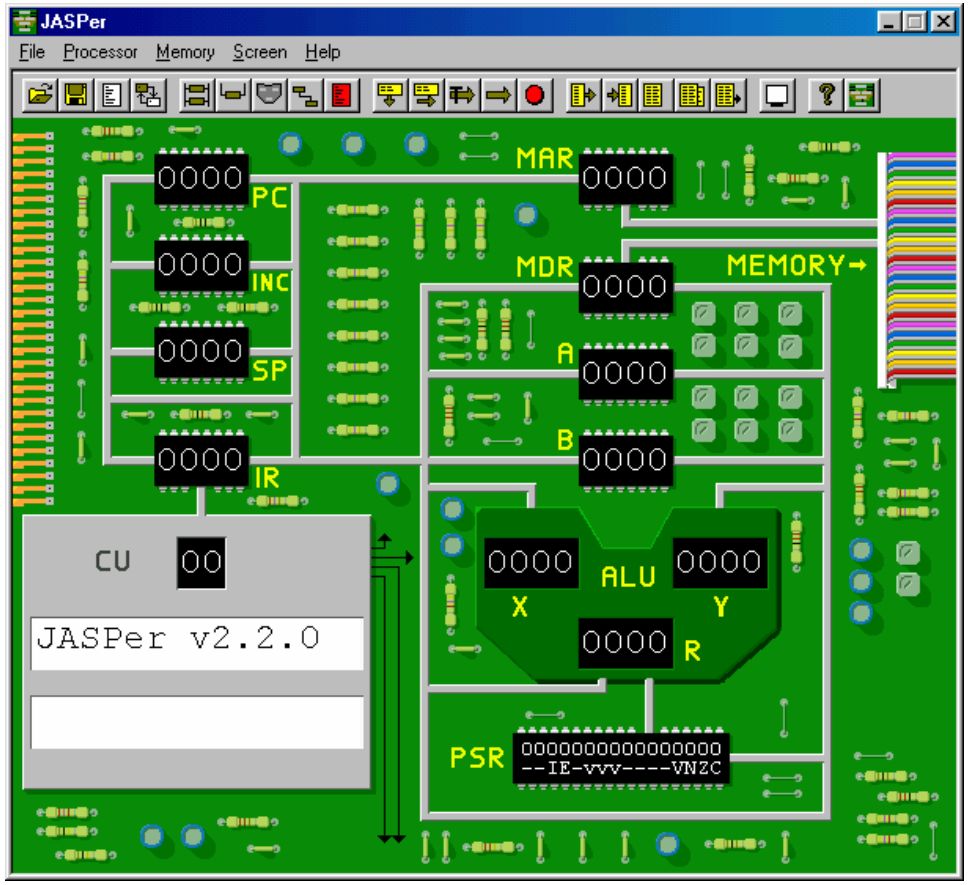


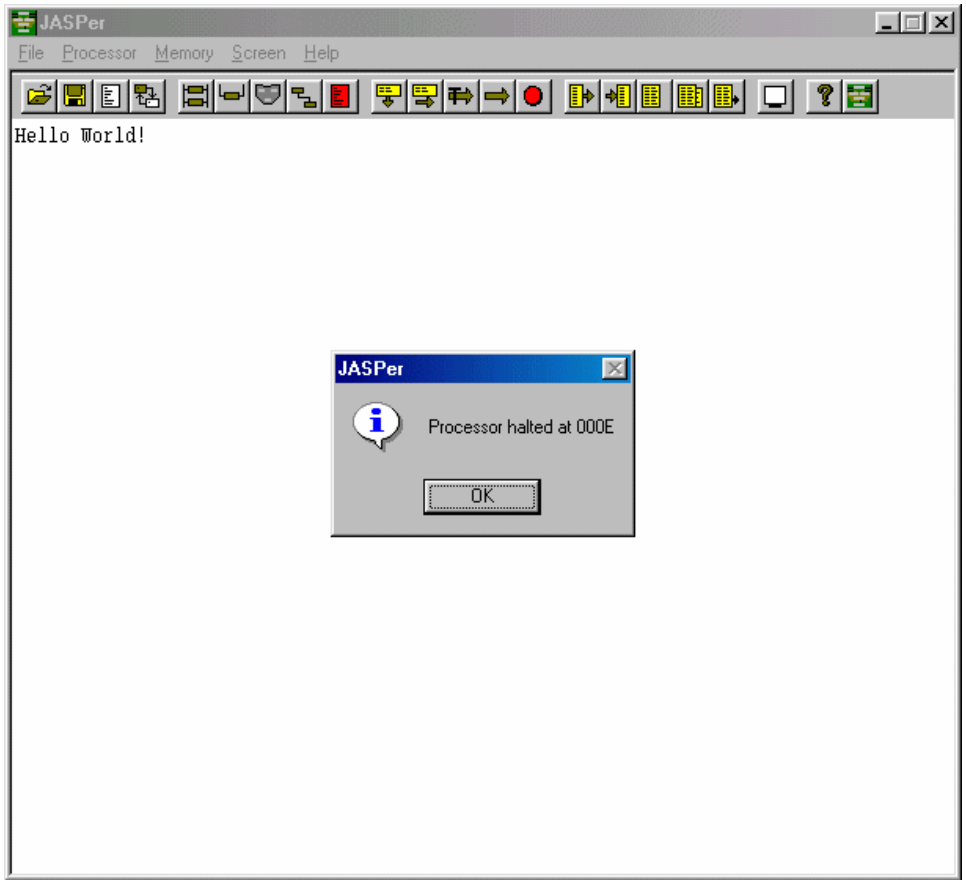
Introducing The Processor

- To build our simple processor we need the following components:
  - Some *Registers* - a register is a store where we can place one piece of data;
  - An *Arithmetic Logic Unit*, or *ALU* - a very basic calculator for our processor.
  - A *Control Unit*, or *CU* - to run the processor;
  - Some buses - to allow us to move data from one component to another.



Introducing The Processor





Fundamental Concepts I - Data Representation

- We covered:
  - Number representation - decimal, binary, octal, hexadecimal and Binary Coded Decimal (BCD);
  - Conversion between different bases;
  - Binary arithmetic;
  - Signed representations - sign and modulus, 1's complement, 2's complement and floating point;
  - Logic operations - AND, OR and NOT;
  - Data representation - ASCII and Unicode.

Fundamental Concepts I - Data Representation

- The simple rule for obtaining the 2's complement representation of the negative of a number is
  - Flip the bits
  - Add 1

		Decimal Representation
0 0 0 0 0 1 1 1		+7
1 1 1 1 1 0 0 0	Flip the bits	
1	Add 1	
1 1 1 1 1 0 0 1	Result represents –7	

Fundamental Concepts I - Data Representation

- Now we know how to figure out the representation of a negative number, let's try some arithmetic

									Decimal Representation
	0	0	0	0	0	1	0	1	+5
+	1	1	1	1	1	1	0	1	−3
	0	0	0	0	0	0	1	0	Result
	(1)	1	1	1	1	1		1	Carries generated

Discard final carry

Fundamental Concepts I - Data Representation

		Decimal Representation
0 1 0 1 1 0 0 0	+	+88
0 0 1 0 1 0 0 1		+41
1 0 0 0 0 0 0 1		Initial result
1 1 1 1	←	Carries generated
1 0 0 0 0 0 0 1		Initial result is negative
0 1 1 1 1 1 1 0		Flip the bits
1		Add 1
0 1 1 1 1 1 1 1		Result is therefore –127 ????

Fundamental Concepts I - Data Representation

0 0 1 0 0 1 1 **1**	X
0 0 1 0 1 1 1 **0**	Y
0 0 1 0 0 1 1 **0**	X AND Y

0 0 1 0 0 1 1 **1**	X
0 0 1 0 1 1 1 **0**	Y
0 0 1 0 1 1 1 **1**	X OR Y

0 0 1 0 0 1 1 **1**	X
1 1 0 1 1 0 0 **0**	NOT X

Fundamental Concepts I - Data Representation

- ASCII
  - For example, the ASCII code for the letter ‘R’ is found as follows:
    - The column that ‘R’ is in is labelled with the hexadecimal digit 5;
    - The row that ‘R’ is in is labelled with the hexadecimal digit 2;
    - This produces the hexadecimal value 52_{16}.
  - ASCII is being replaced by 16-bit Unicode.

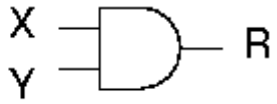
Low Hexadecimal Digit \ High Hexadecimal Digit	0	1	2	3	4	5	6	7
0	NUL	DCL		0	@	P	‘	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	’	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	–	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

Fundamental Concepts II - Digital Electronic Circuits

- We covered:
  - Gate logic - AND, OR and NOT;
  - How to build circuits with gates;
  - Modelling circuits with truth tables;
  - Boolean algebra, including De Morgan's laws.

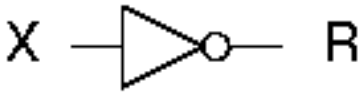
Fundamental Concepts II - Digital Electronic Circuits

X	Y	R
0	0	0
0	1	0
1	0	0
1	1	1



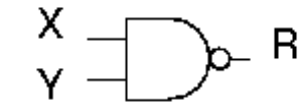
X	Y	R
0	0	0
0	1	1
1	0	1
1	1	1

X	R
0	1
1	0

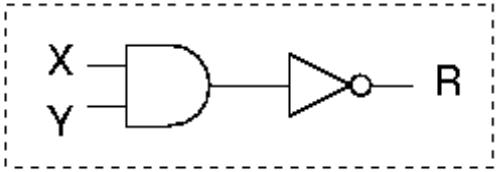


Fundamental Concepts II - Digital Electronic Circuits

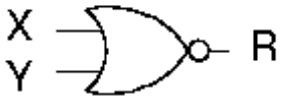
X	Y	R
0	0	1
0	1	1
1	0	1
1	1	0



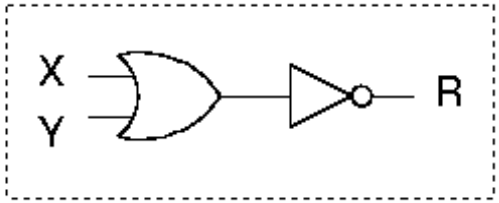
NAND



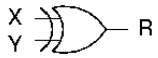
X	Y	R
0	0	1
0	1	0
1	0	0
1	1	0



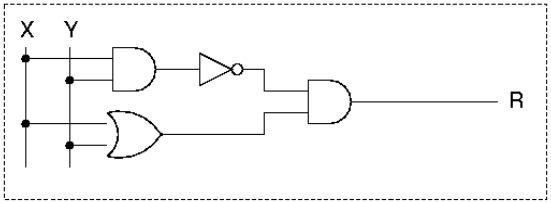
NOR



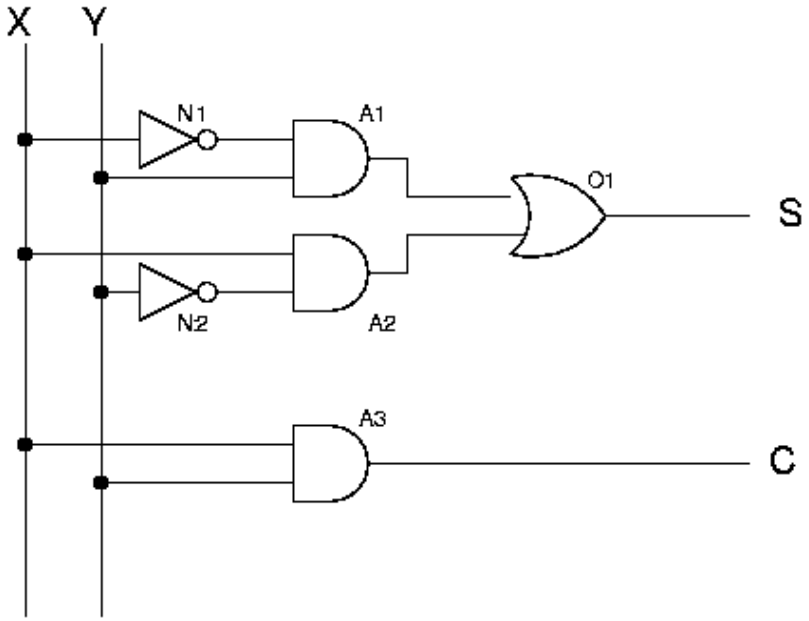
X	Y	R
0	0	0
0	1	1
1	0	1
1	1	0



XOR



Fundamental Concepts II - Digital Electronic Circuits



X	Y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- A circuit diagram can be derived from a truth table

Fundamental Concepts II - Digital Electronic Circuits

- **Boolean Algebra**
  - The set of rules that we can make use of are known as the identities of boolean algebra.
- Each identity (apart from the absorbtion and double complement) has two forms, one for the AND form, and the other for the OR form.

Law	Identity
Absorbtion	$x \cdot (x + y) = x$
Associative	$x + (y + z) = (x + y) + z$ $x \cdot (y \cdot z) = (x \cdot y) \cdot z$
Complement	$x + \bar{x} = 1$ $x \cdot \bar{x} = 0$
Commutative	$x + y = y + x$ $x \cdot y = y \cdot x$
De Morgan's Laws	$\overline{(x \cdot y)} = \bar{x} + \bar{y}$ $\overline{(x + y)} = \bar{x} \cdot \bar{y}$
Distributive	$x + (y \cdot z) = (x + y) \cdot (x + z)$ $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
Dominance	$x + 1 = 1$ $x \cdot 0 = 0$
Double Complement	$\overline{(\bar{x})} = x$
Idempotent	$x + x = x$ $x \cdot x = x$
Identity	$x + 0 = x$ $x \cdot 1 = x$

Fundamental Concepts II - Digital Electronic Circuits

- Boolean Algrebra
  - De Morgan's Laws can help us in the creation of *efficient* digital circuits.

$$\overline{(x \cdot y)} = \bar{x} + \bar{y}$$
$$\overline{(x + y)} = \bar{x} \cdot \bar{y}$$

Gate	Number of transistors
AND	3
OR	3
NOT	1
NAND	2
NOR	2

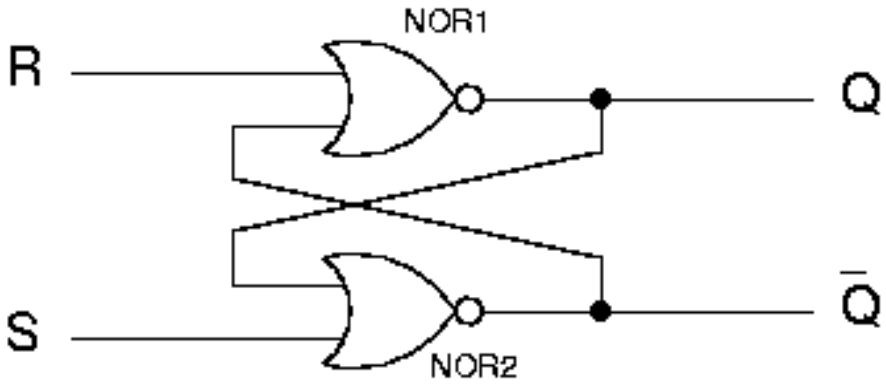
Registers

- We covered:
  - Bistables - the RS latch, the D latch and the D flip-flop;
  - How to build a register;
  - Tri-state logic;
  - The concept of a clock and a clock cycle.

Registers

The RS Latch

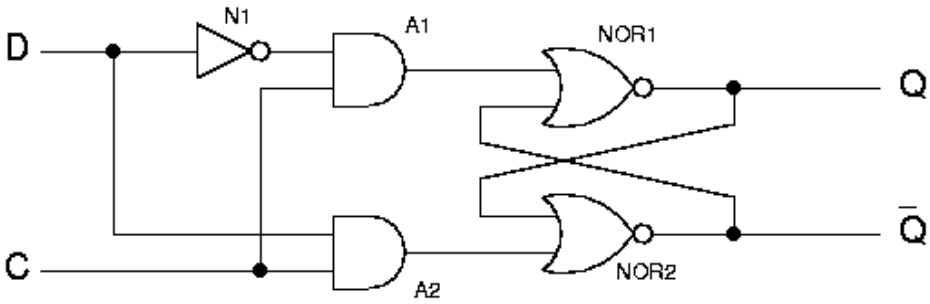
R	S	Description
0	0	Q is left at what it was previously set to
0	1	$Q = 1$
1	0	$Q = 0$
1	1	$Q = \bar{Q} = 0$



Registers

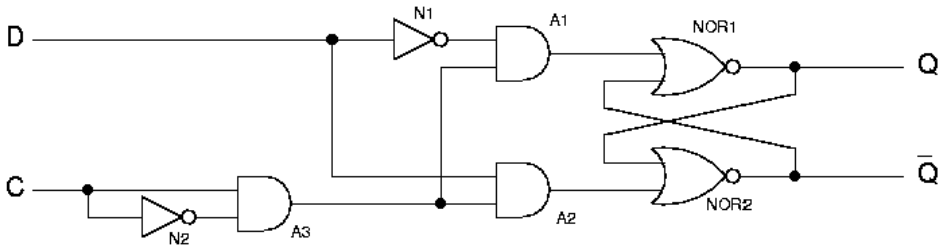
The D Latch

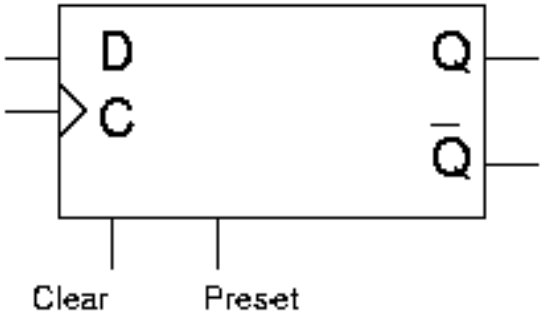
D	C	Description
0	0	Q is left at what it was previously set to.
0	1	Q = 0
1	0	Q is left at what it was previously set to.
1	1	Q = 1



Registers

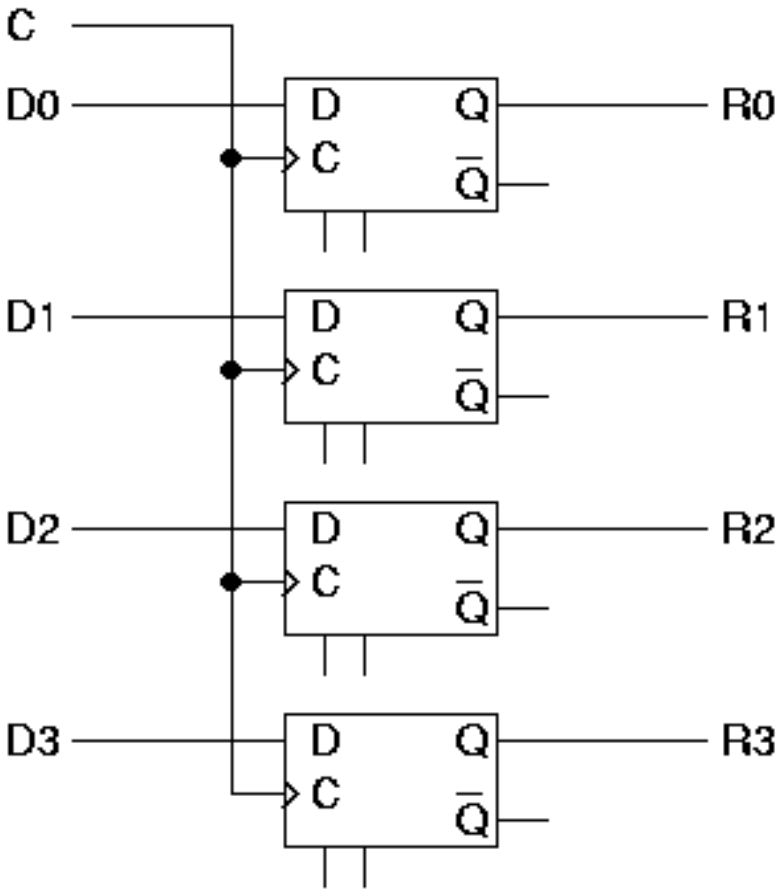
The D Flip-flop



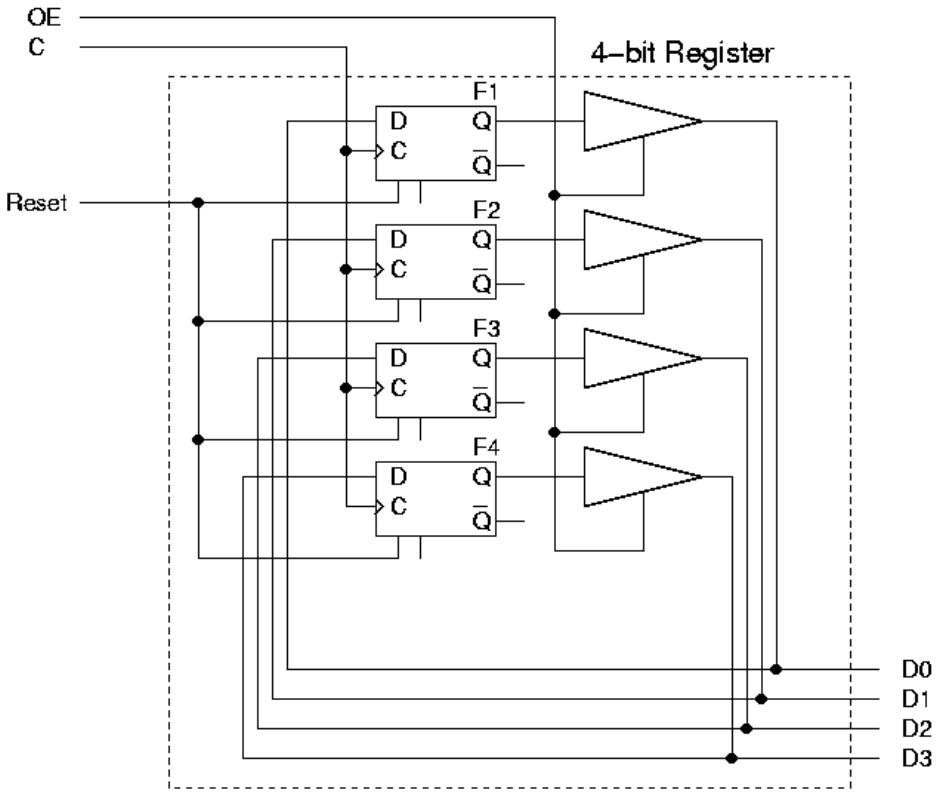


Registers

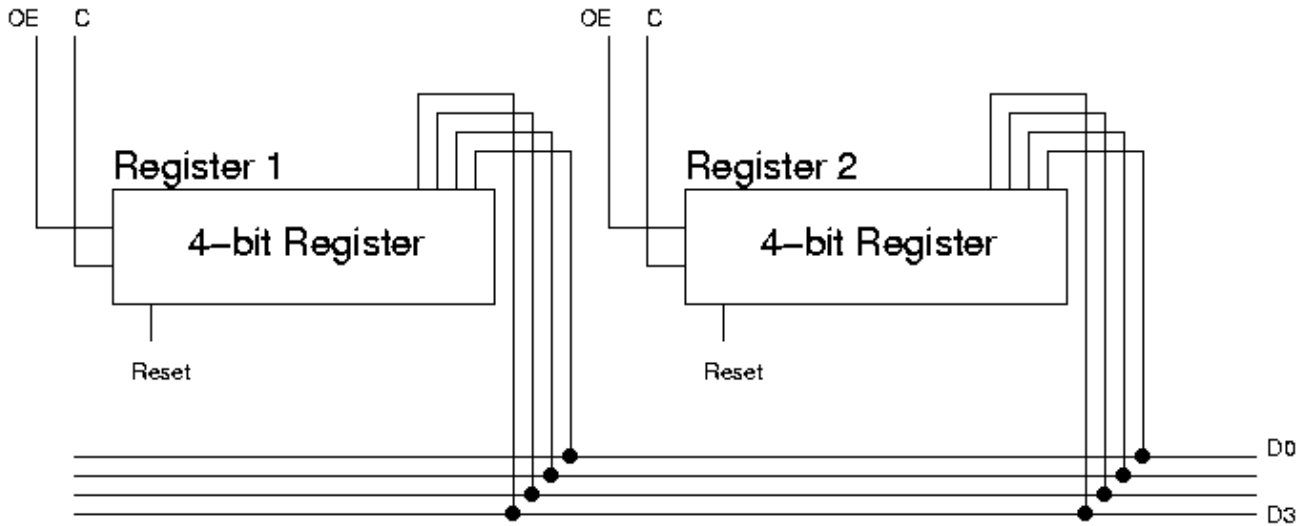
A 4-bit register



A 4-bit register attached to a bus



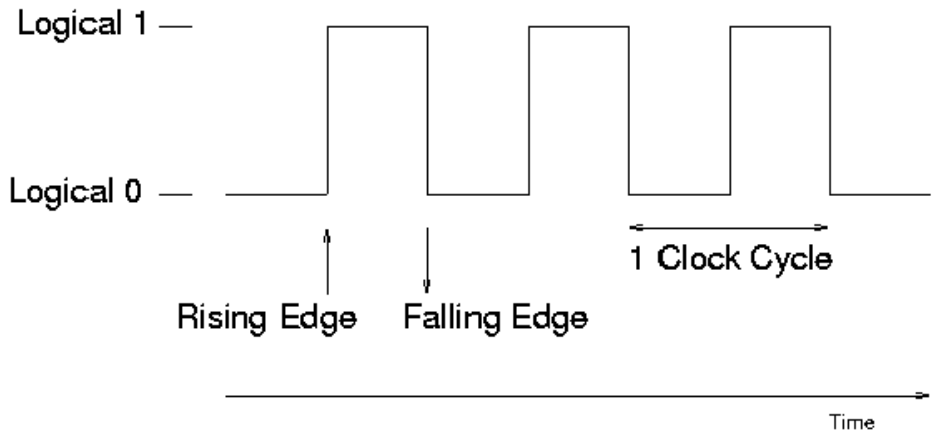
Registers



- To move a bit pattern from register 1 to register 2 we would do as before:
  - Set register 1 OE to 1 (bit pattern now on the bus);
  - Clock register 2 (for register 2 to take the bit pattern from the bus);
  - Set register 1 OE to 0;

Registers

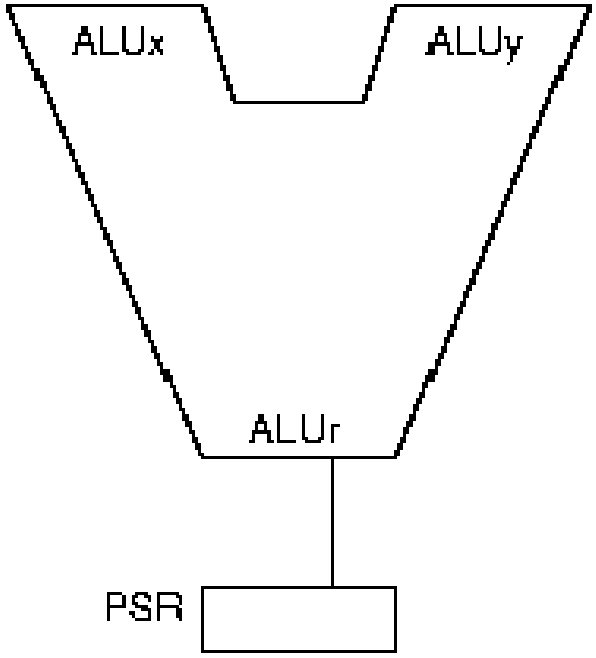
- The Clock cycle
- There are effectively four different types of trigger. These are:
  - 1 triggered - when the clock signal becomes 1;
  - 0 triggered - when the clock signal becomes 0;
  - Positive edge triggered - when the clock signal changes from a 0 to a 1;
  - Negative edge triggered - when the clock signal changes from a 1 to a 0.

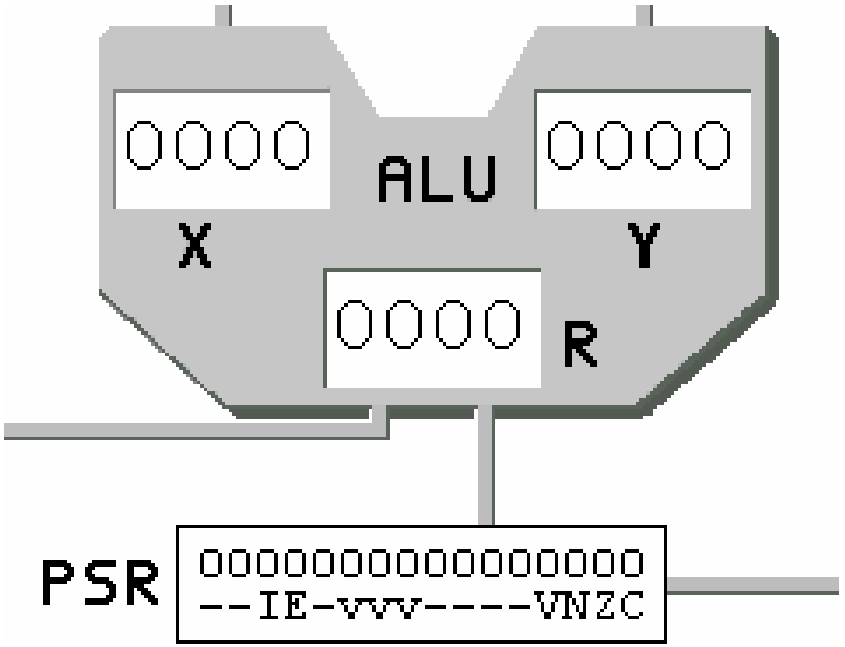


The ALU

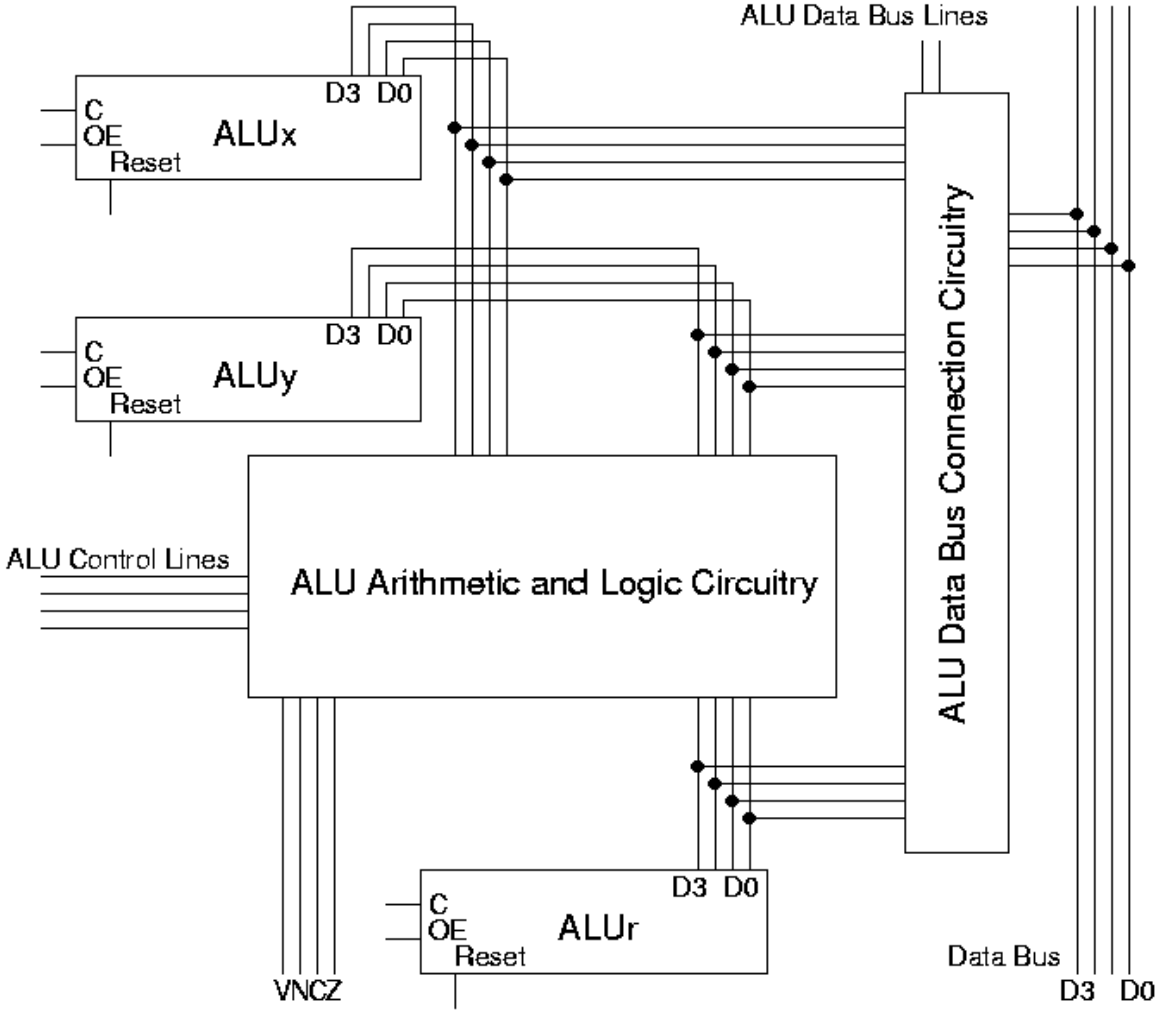
- We covered:
  - The role of the ALU and PSR within the processor;
  - The control circuitry of the ALU;
  - Adder circuits - the half adder and the full adder;
  - Building circuits to demonstrate the functionality of the ALU – the ADD, SL and NEG circuits.

The ALU



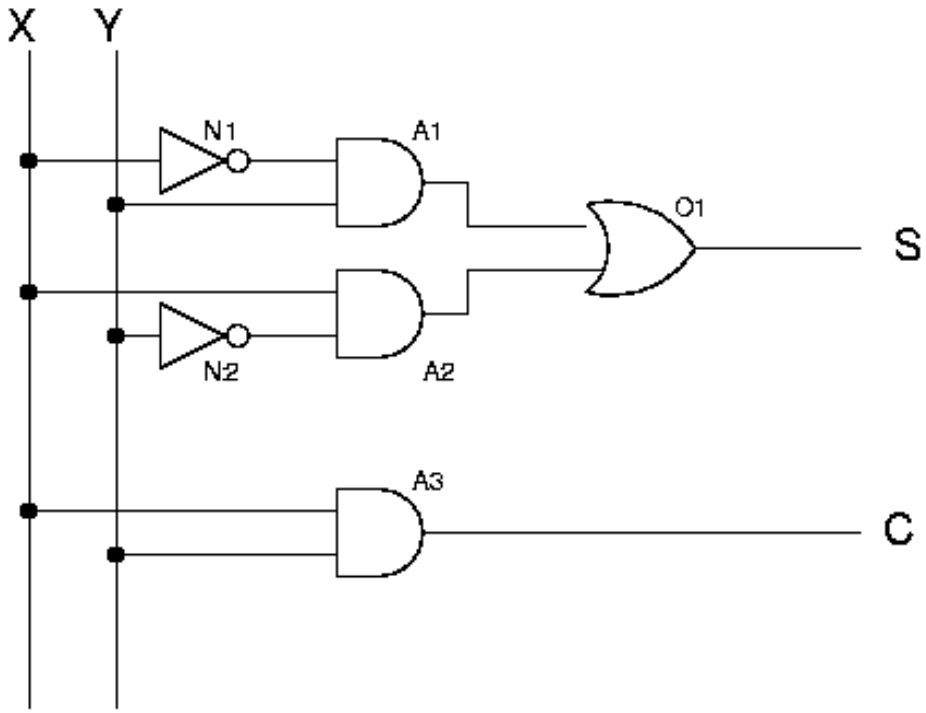


The ALU

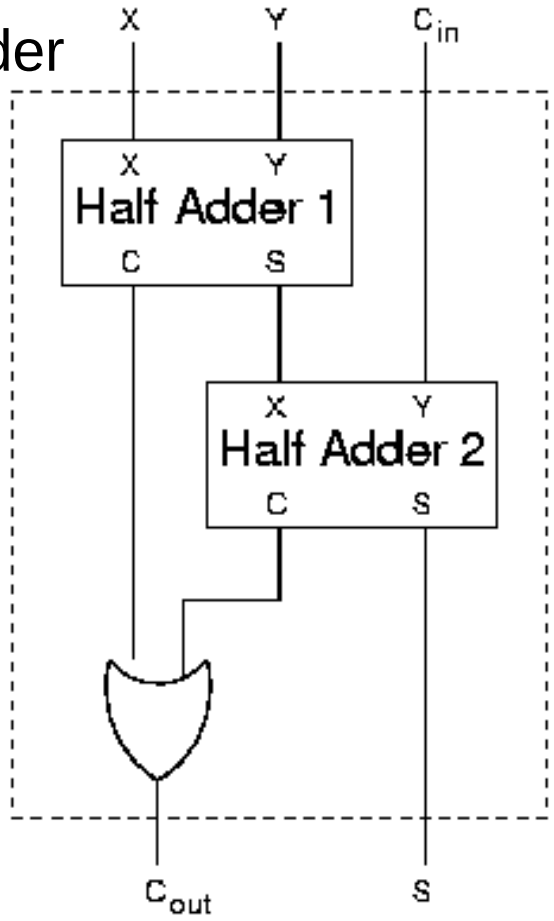


The ALU

Half Adder

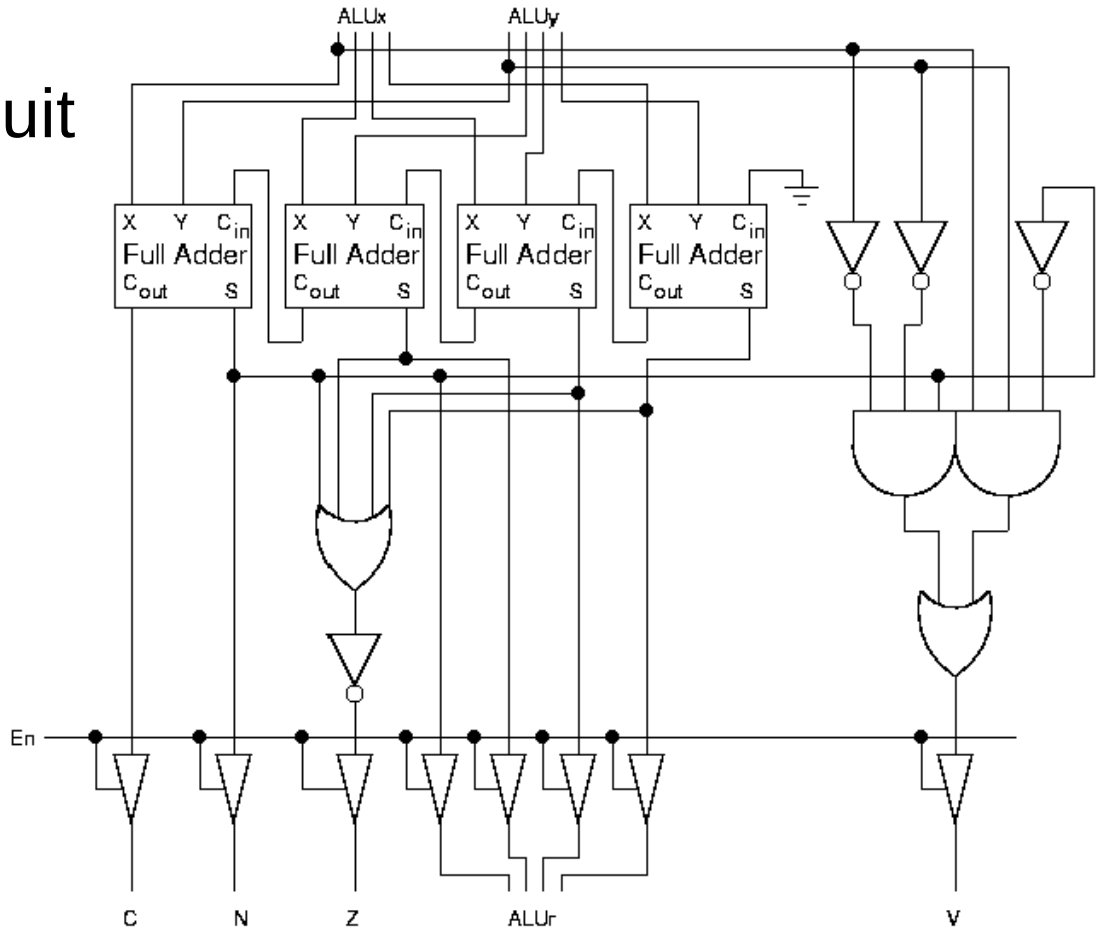


Full Adder



The ALU

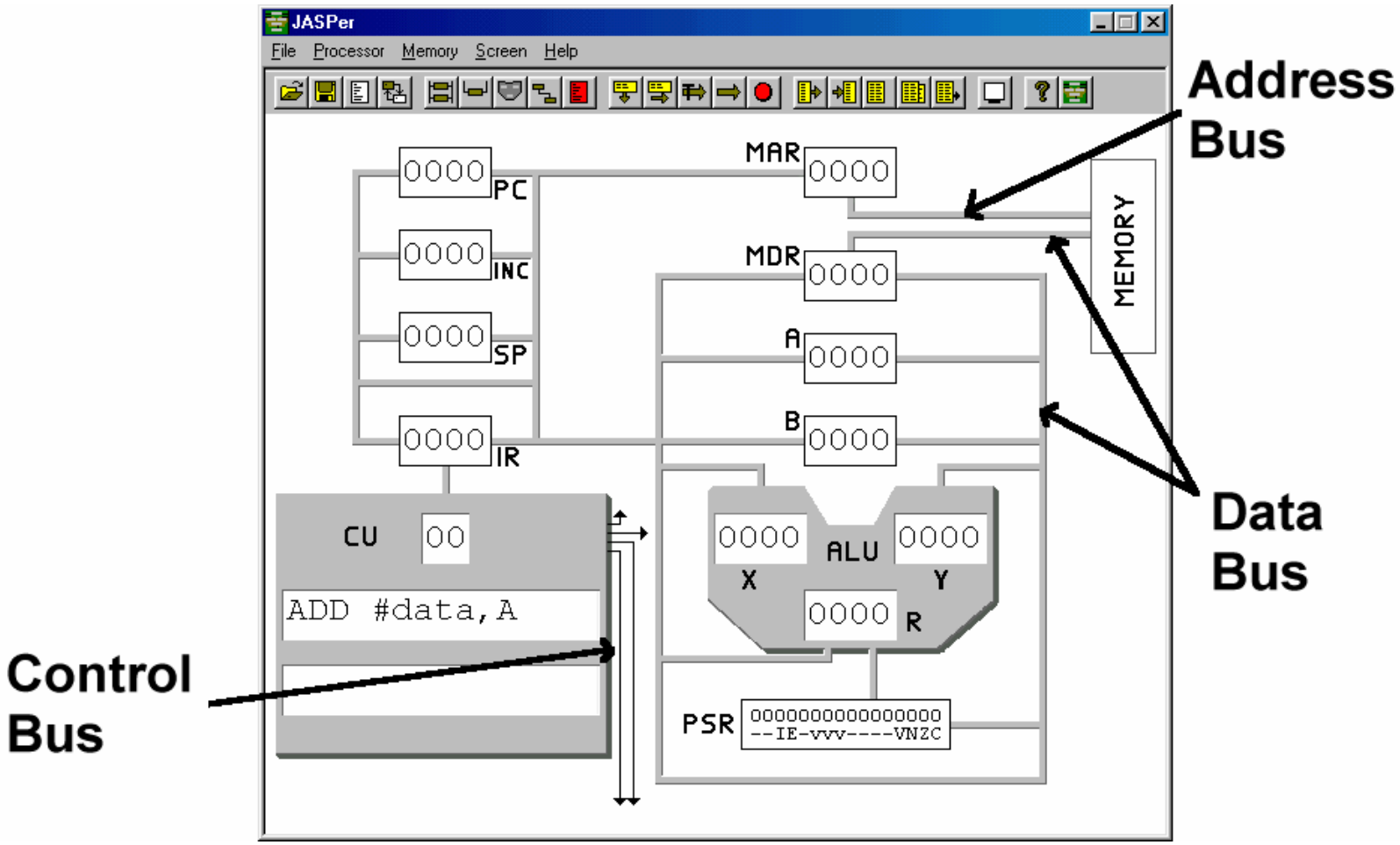
The ADD circuit



Buses

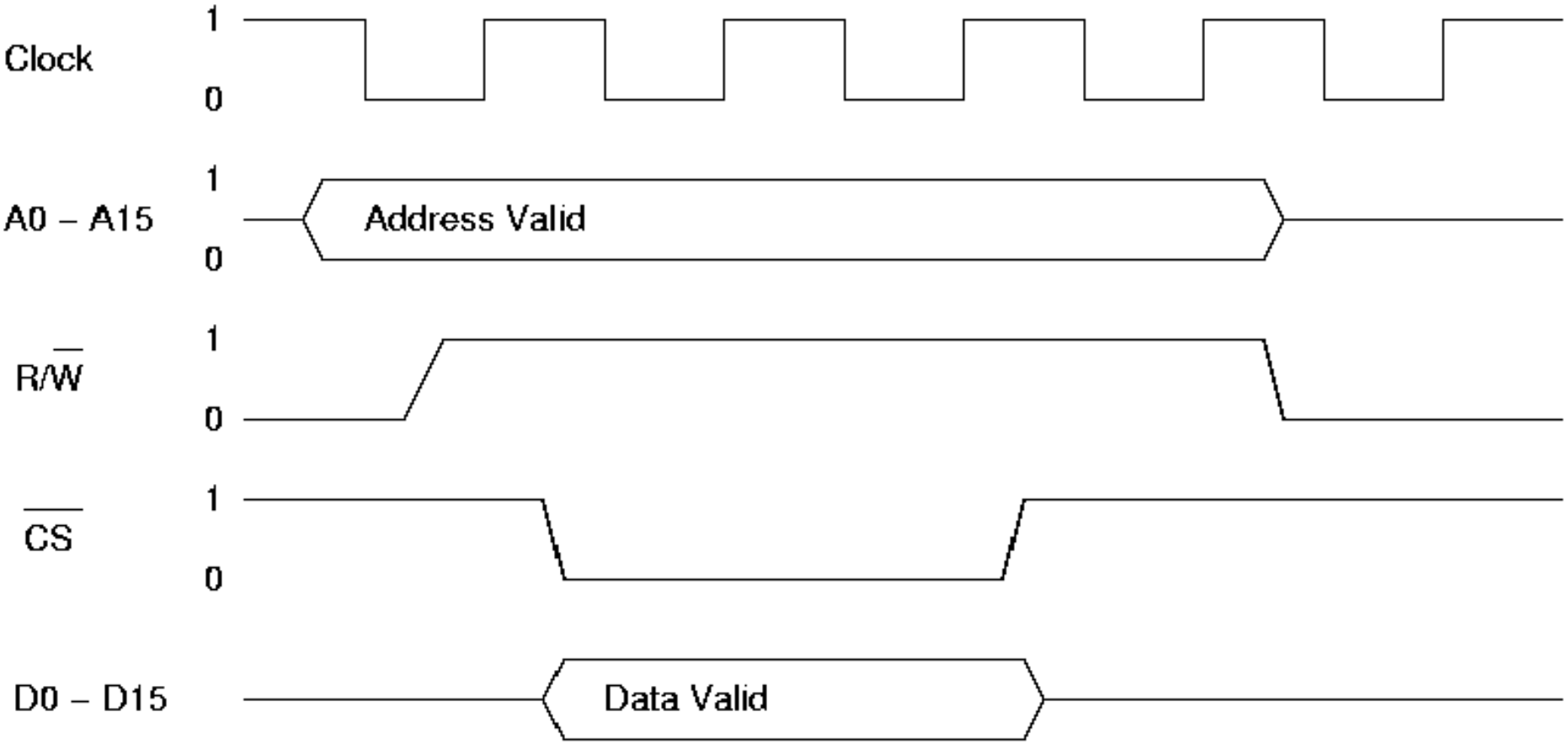
- We covered:
  - Processor buses - the data bus, the address bus and the control bus;
  - Building a bus with gate logic;
  - Timing diagrams.

Buses



Buses

A timing diagram

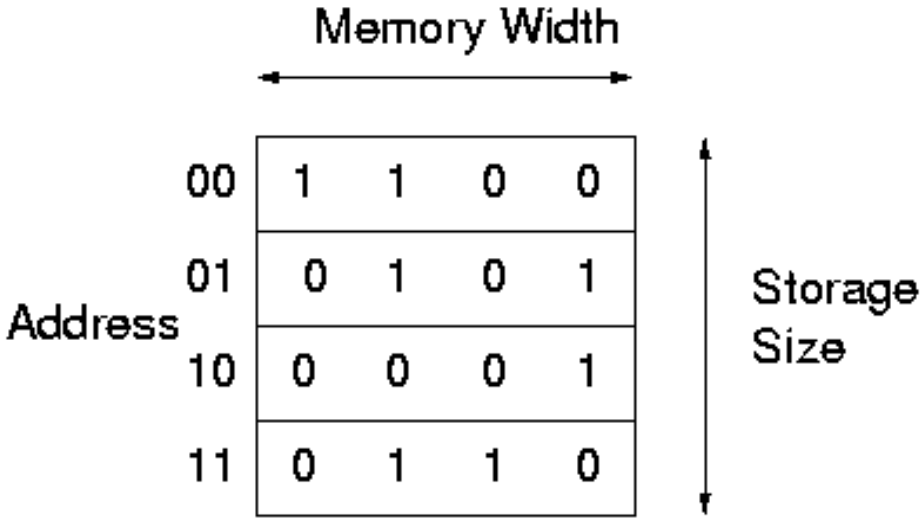


Memory

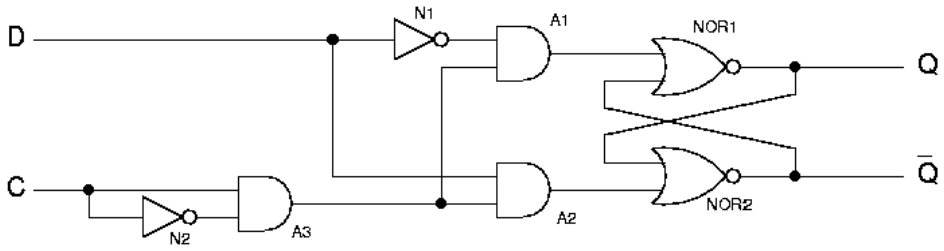
- We covered:
  - The concepts of memory;
  - How to build memory from gate logic;
  - Types of memory;
  - Address decoding strategies;
  - Memory maps.

Memory

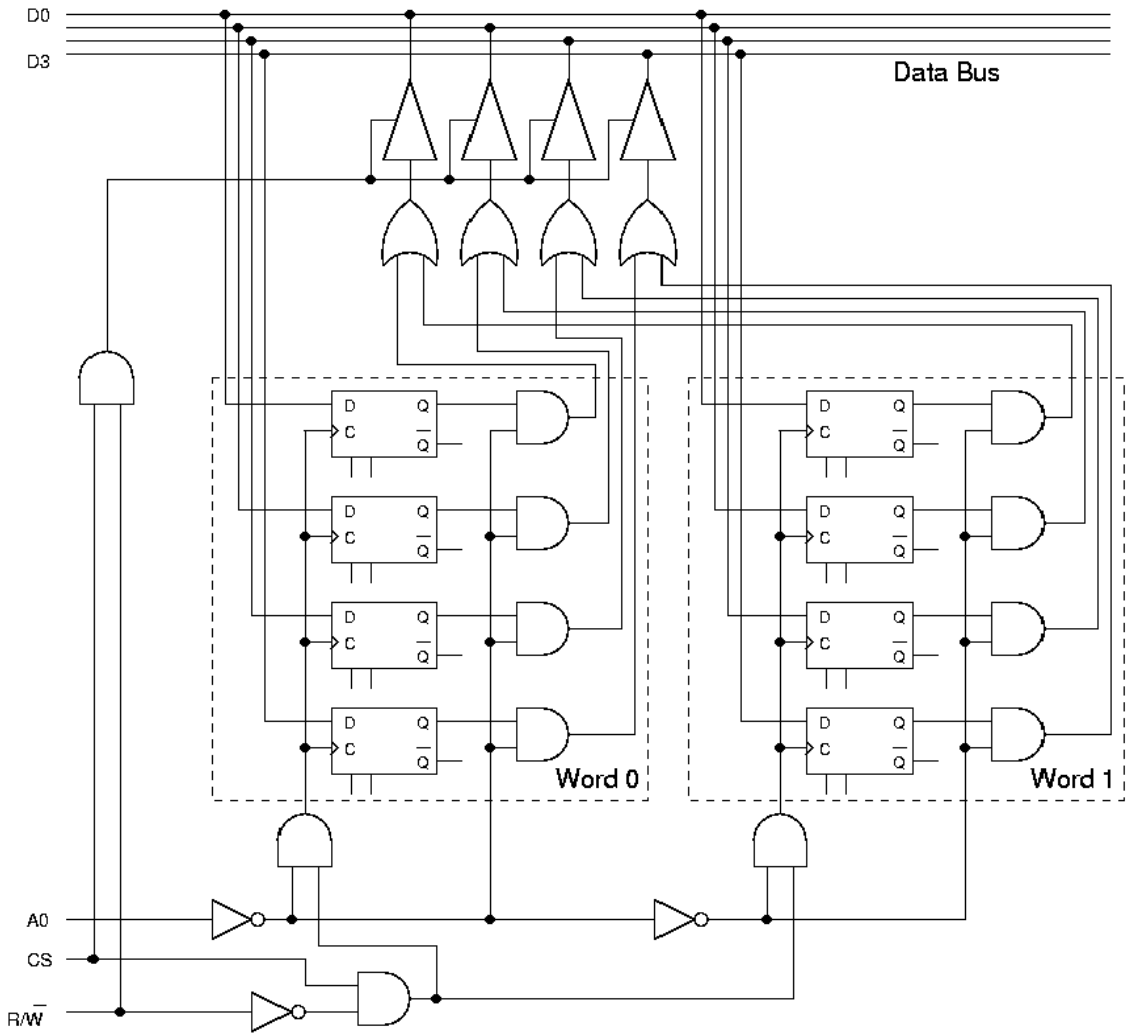
A small memory



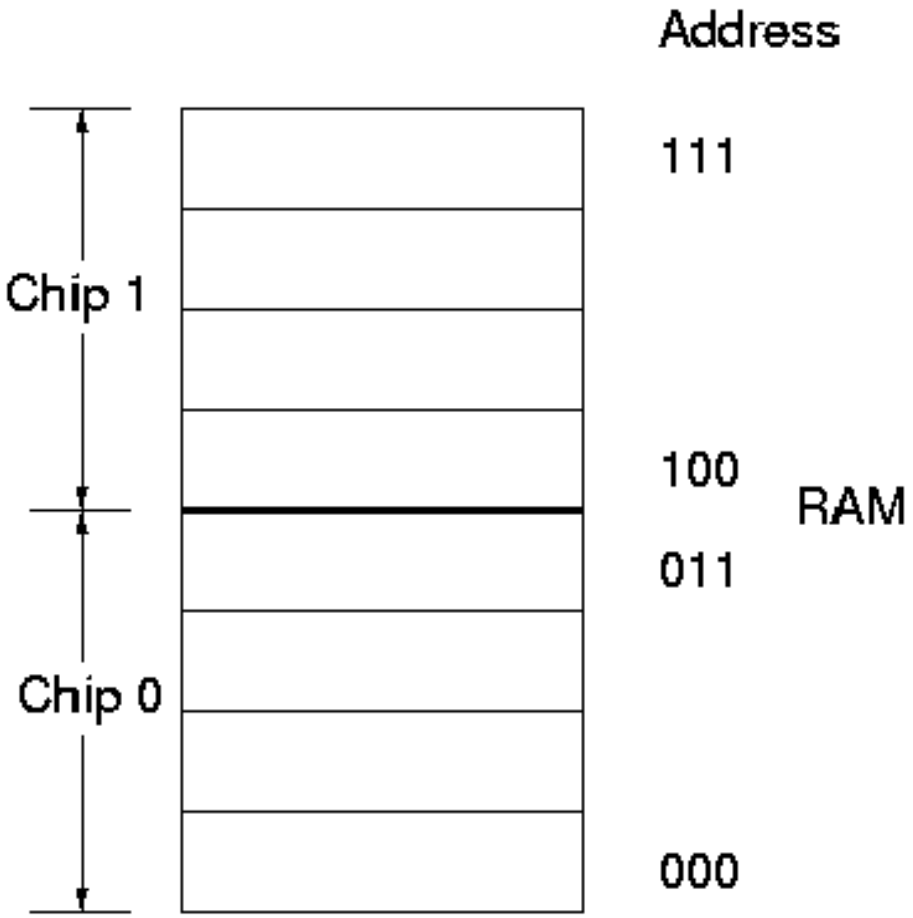
A D flip-flop

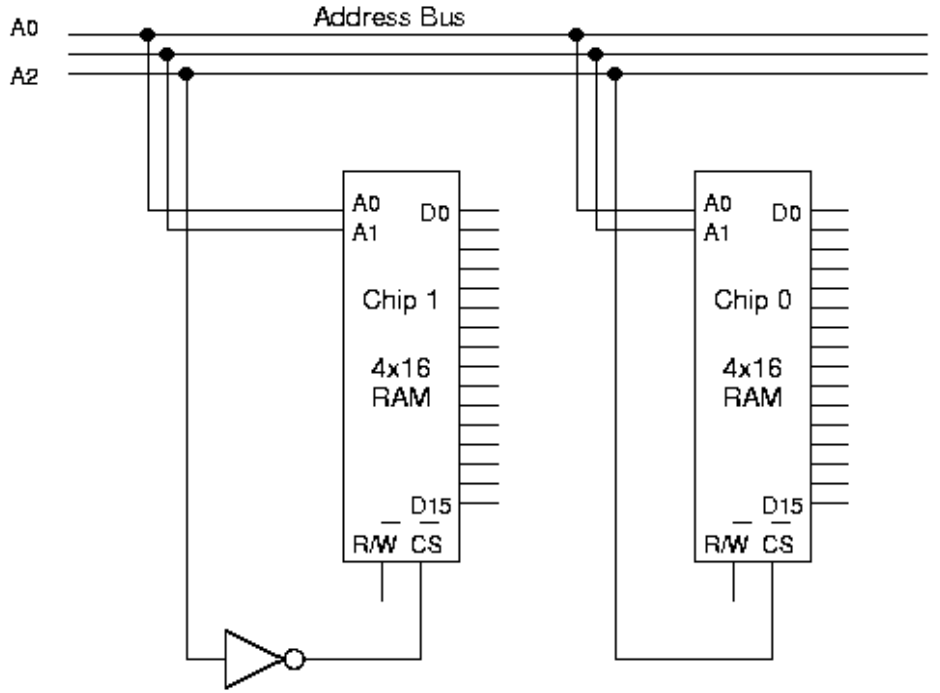


Memory



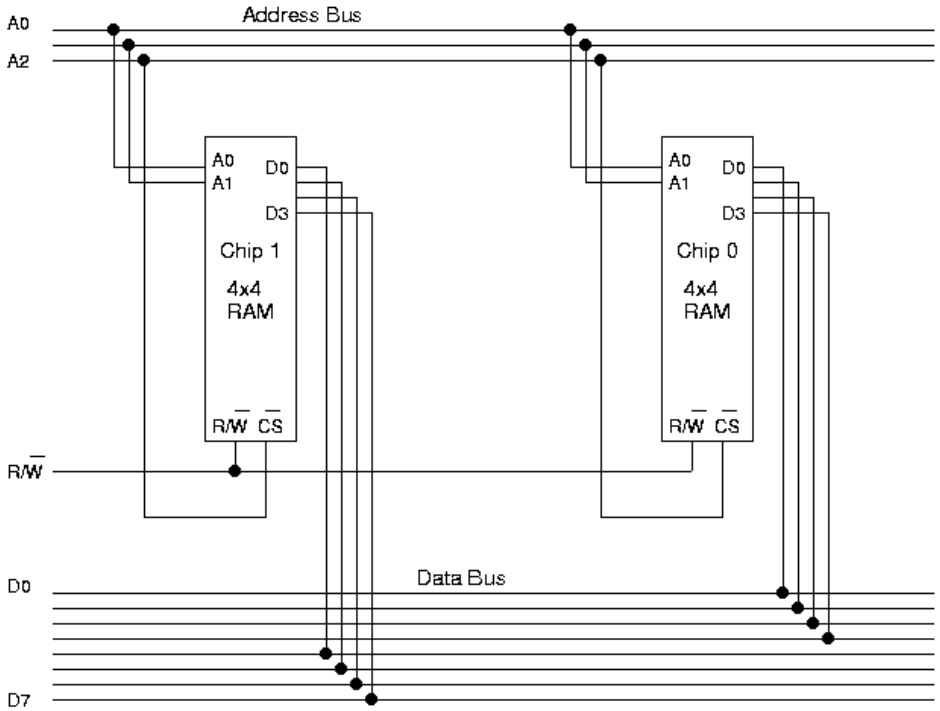
Memory



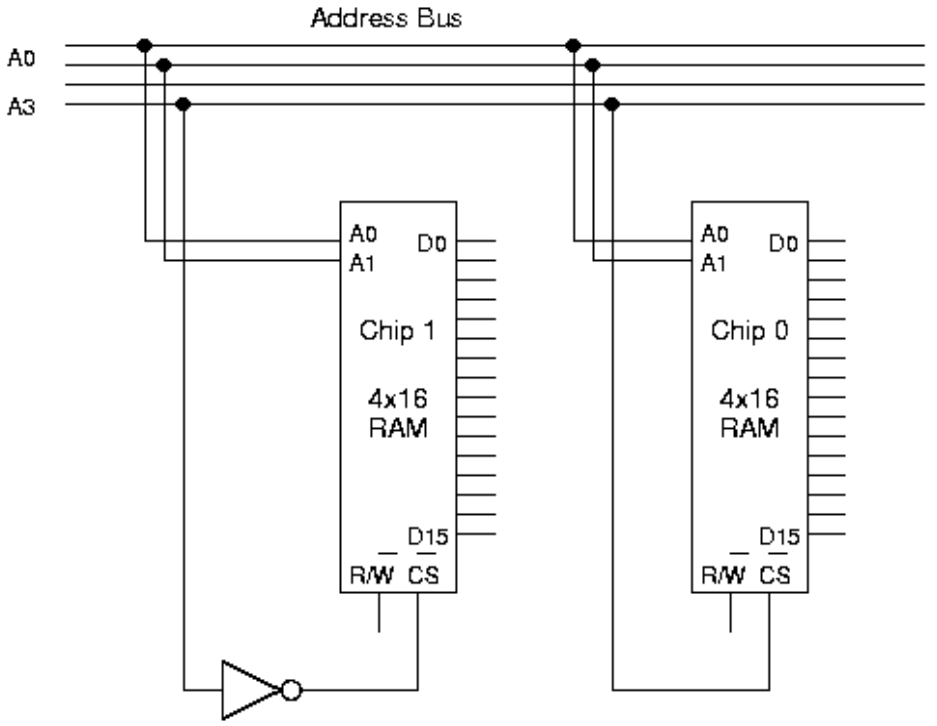


Memory

Building wider memories



Partial address mapping



Memory

JASPer memory