Lecture 16

Random Access Memory and the Fetch Cycle

Random Access Memory

We continue to put together a computer system, and in this lecture add the memory.

We will consider first the implementation of random access memory, and then turn to how it is used by the central processor.

One Bit Memory

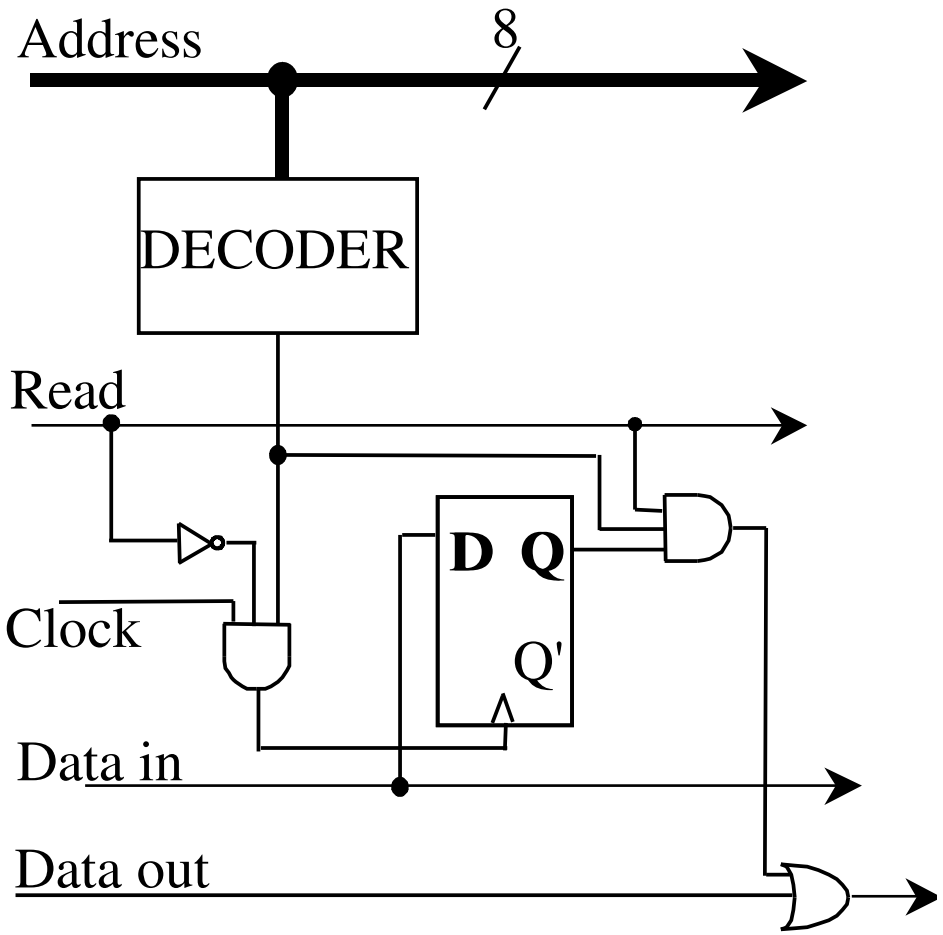
A single D type flip flop is a one bit memory

To use it as such we need to give it an address

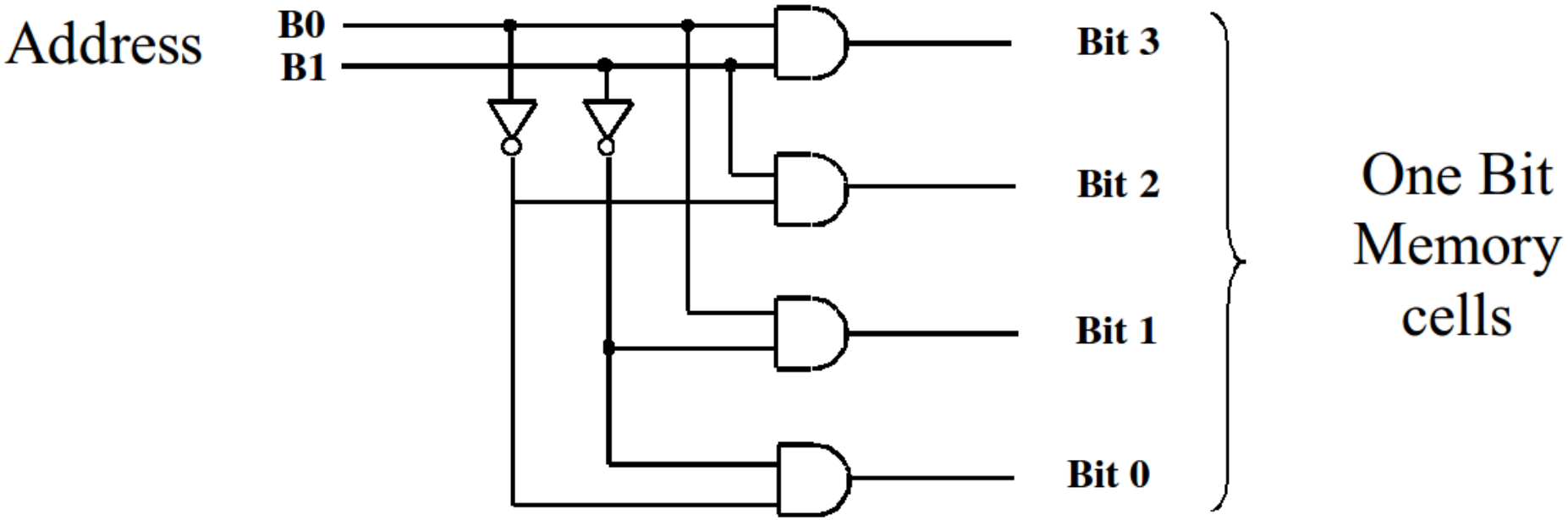
An address is simply a binary number

A binary number can be uniquely identified by a decoder.

One bit static RAM



A decoder is a binary to unary converter



The circuit is asymmetric

For reading it is merely a combinatorial circuit

for writing it is a sequential circuit, the address and data must be present and correct when the clock pulse sets the flip flop.

RAM circuits conforming to this pattern are called static RAMs, and are used in special applications.

Buses

Buses are data highways which are common to many circuits.

The address lines that go to the memory are referred to as the address bus.

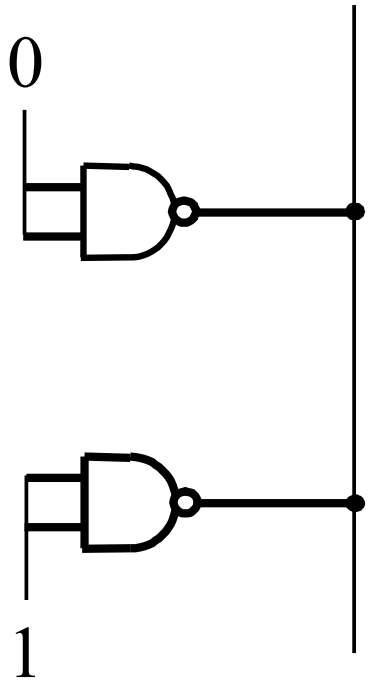
The read, data in and data out lines lines go to every cell, and could also be called a bus.

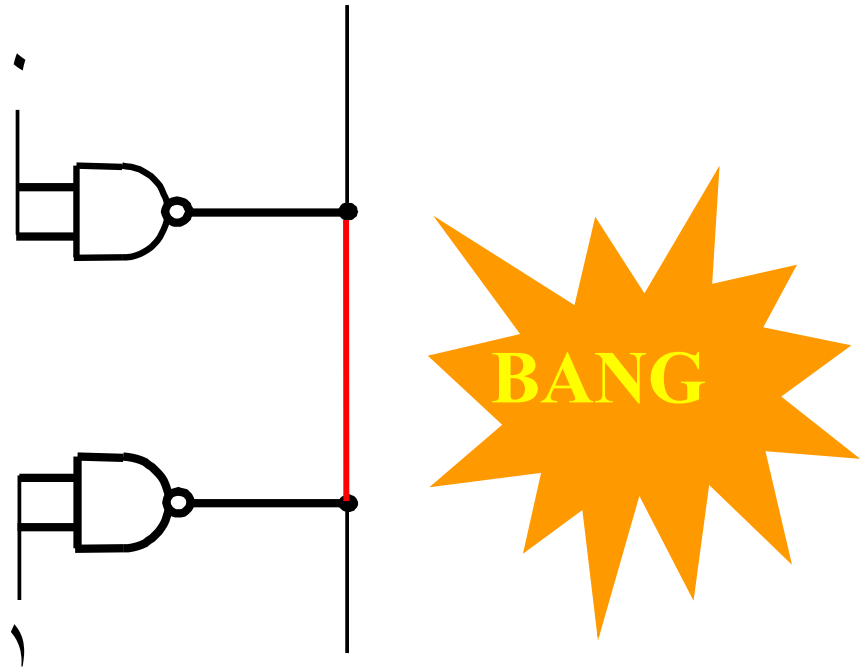
Bi-Directional data buses

The data in and data out lines are never both used at the same time.

It is convenient to use just one line as this reduces the size and complexity of the memory circuit.

However, to make the data line bi-directional we need to feed it from more than one place and we cannot do this with AND or NAND gates

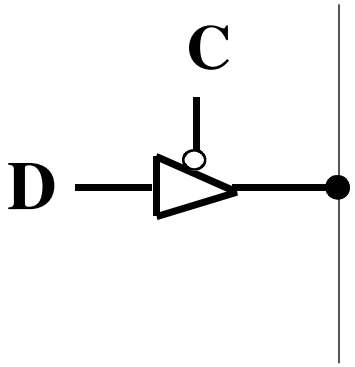

0
1


0
1
BANG

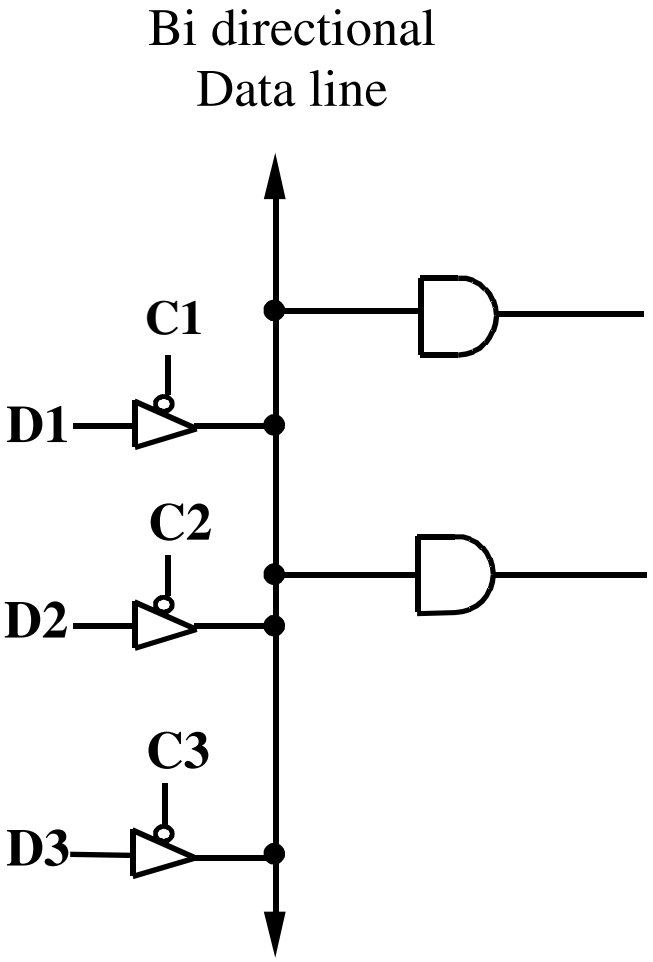
Tri-State Buffer

If the control line, C, is set to zero the output follows the input exactly

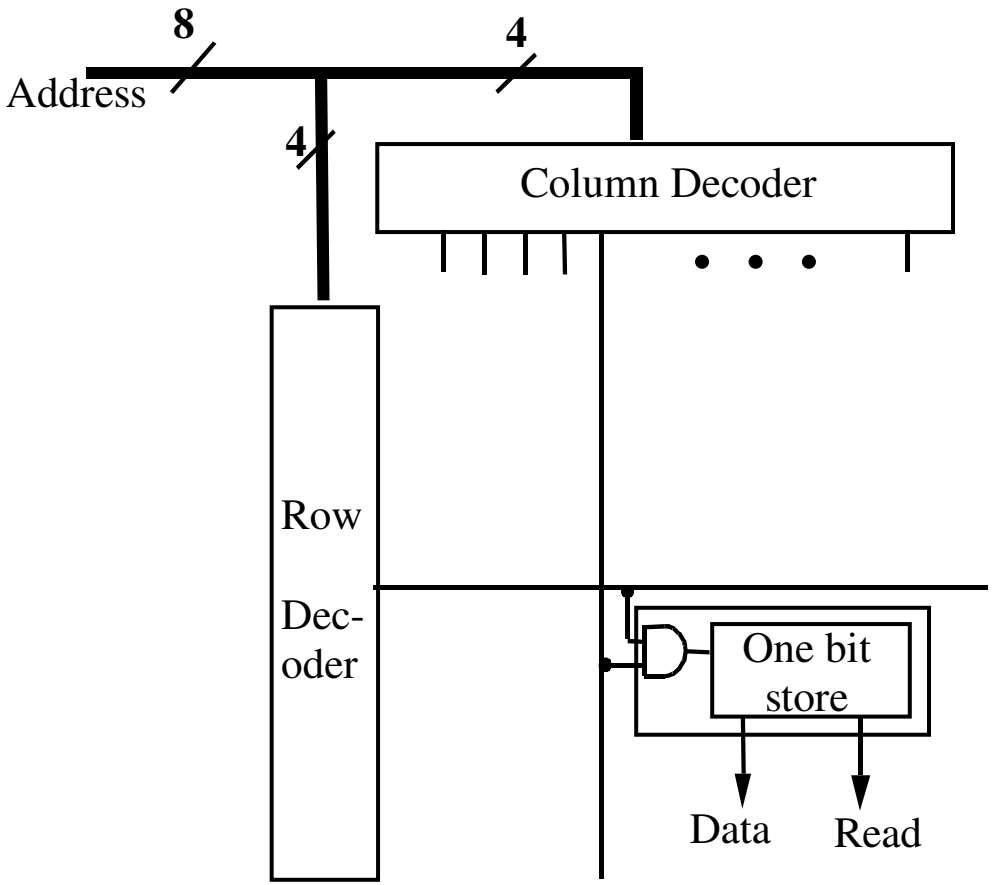
If C is set to 1 the output is neither zero nor one, but is effectively disconnected from the data line.



Interfacing with tri-state buffers



Row/Column organisation of RAM



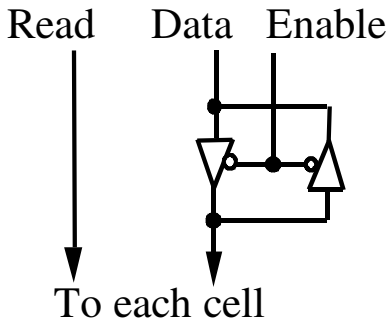


Diagram 16.3
Layout of RAM

Practical RAM circuits

Each one bit memory cell is only enabled when both its row and the column lines are one.

In the case of a 256 bit RAM each decoder transforms a four bit binary number into a sixteen bit unary number.

In the square array of one bit memory cells, there will only ever be one cell for which both the row and the column lines are one.

Practical RAM circuits (again)

Each cell is connected to the same read/write line and data line.

The data line is connected to the outside through a two way tri state buffer, so that unless the chip is enabled no data can pass either in or out.

This enables us to build external decoders for larger capacity RAMs made up of several banks of single chips.

Problem Break

Given a 32 bit address bus:

1. How many bits of RAM could we address?
2. How many bytes of RAM could we address?
3. What size decoder do we need for the row and column decoders?

Answer

With a 32 bit address bus we have 2^{32} possible addresses which is in total 4GBytes.

Each bit of a byte will have the same address so again we can address 4GBytes

The row and column decoders will both be 16 to 64K decoders. $(64K)^2$ = 4 GBytes

Connecting RAM to a processor

Memory Address Register (MAR)

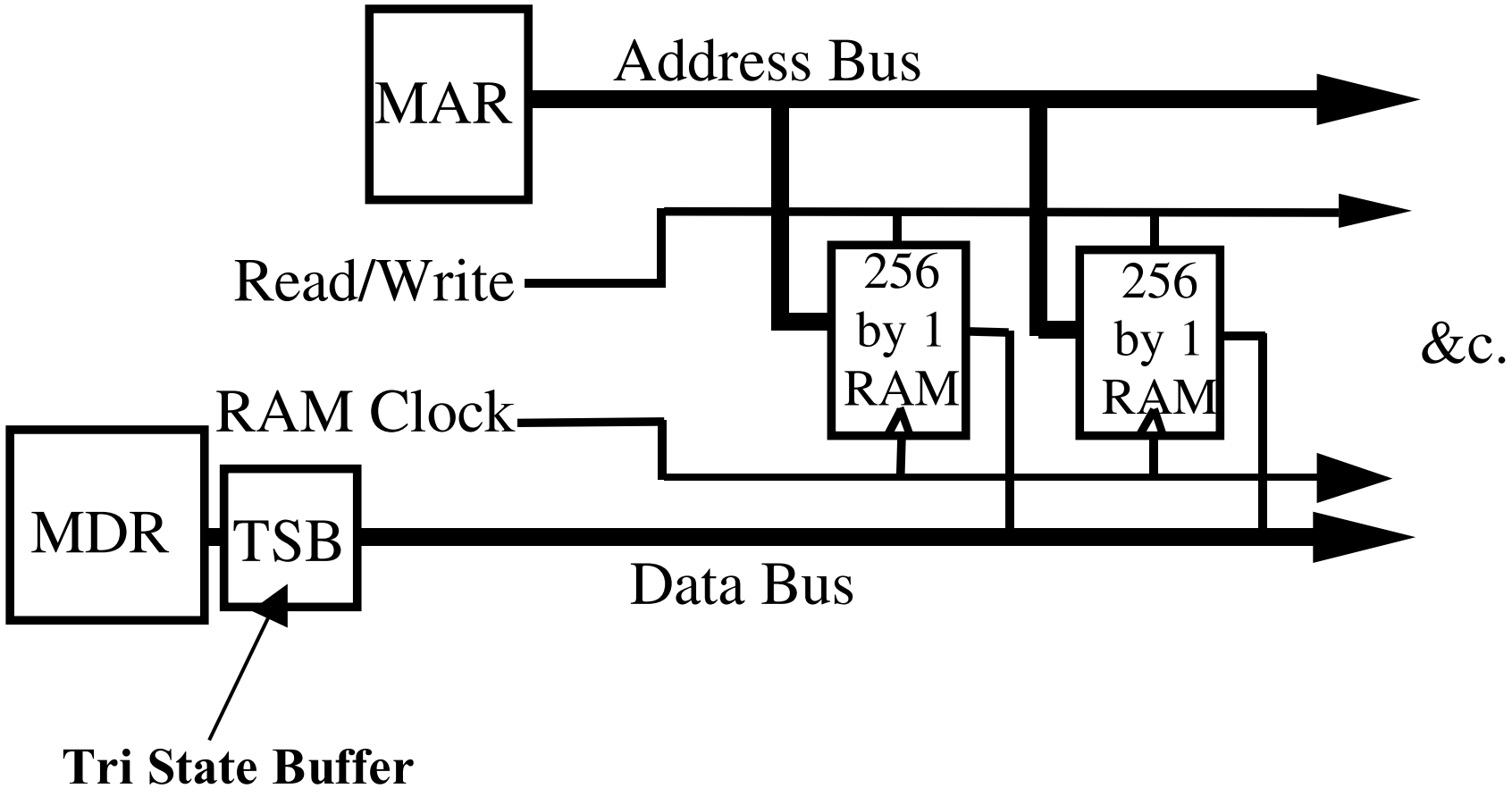
Stores the address in memory for data transfer

Memory Data Register (MDR) or Memory Base Register (MBR).

Stores data to be transferred to memory

(Can be used for other purposes)

Connecting RAM to a processor



Connecting RAM to a processor

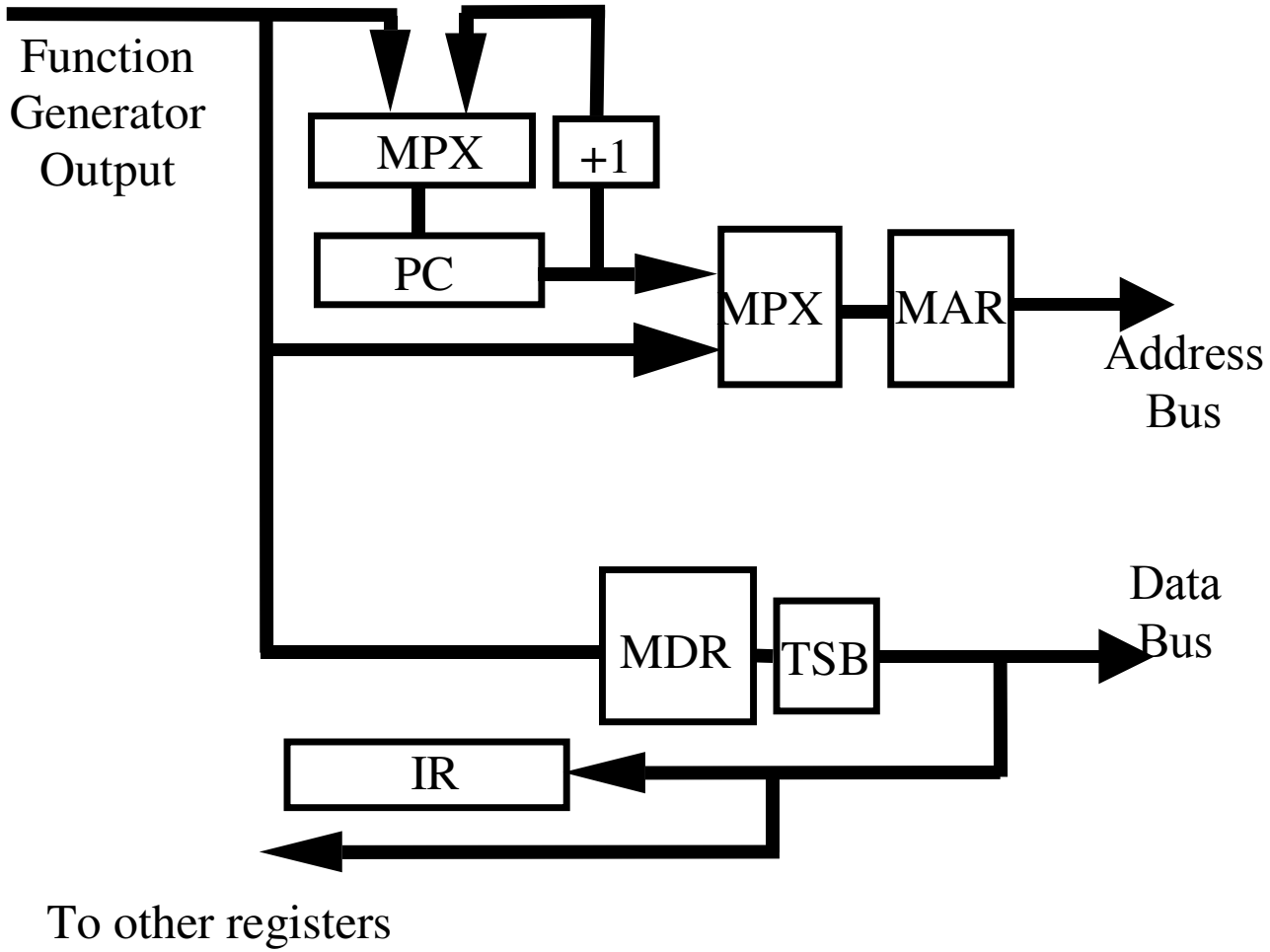
Program Counter (PC)

stores the address of the next program instruction to be executed

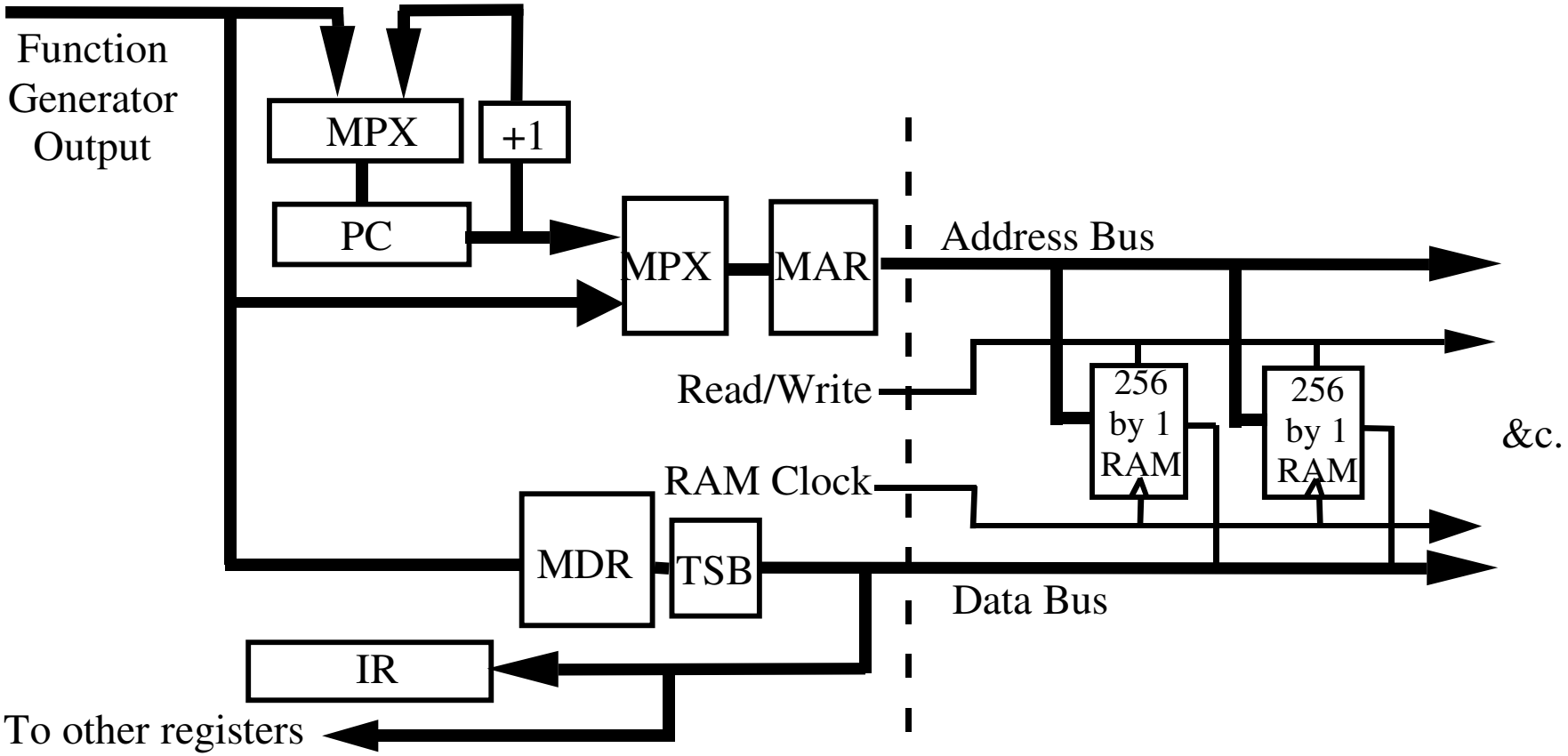
Instruction Register

stores the program instruction being executed.

Connecting RAM to a processor



The whole connection



The Fetch Cycle

The fetch cycle gets the next program instruction from memory into the IR

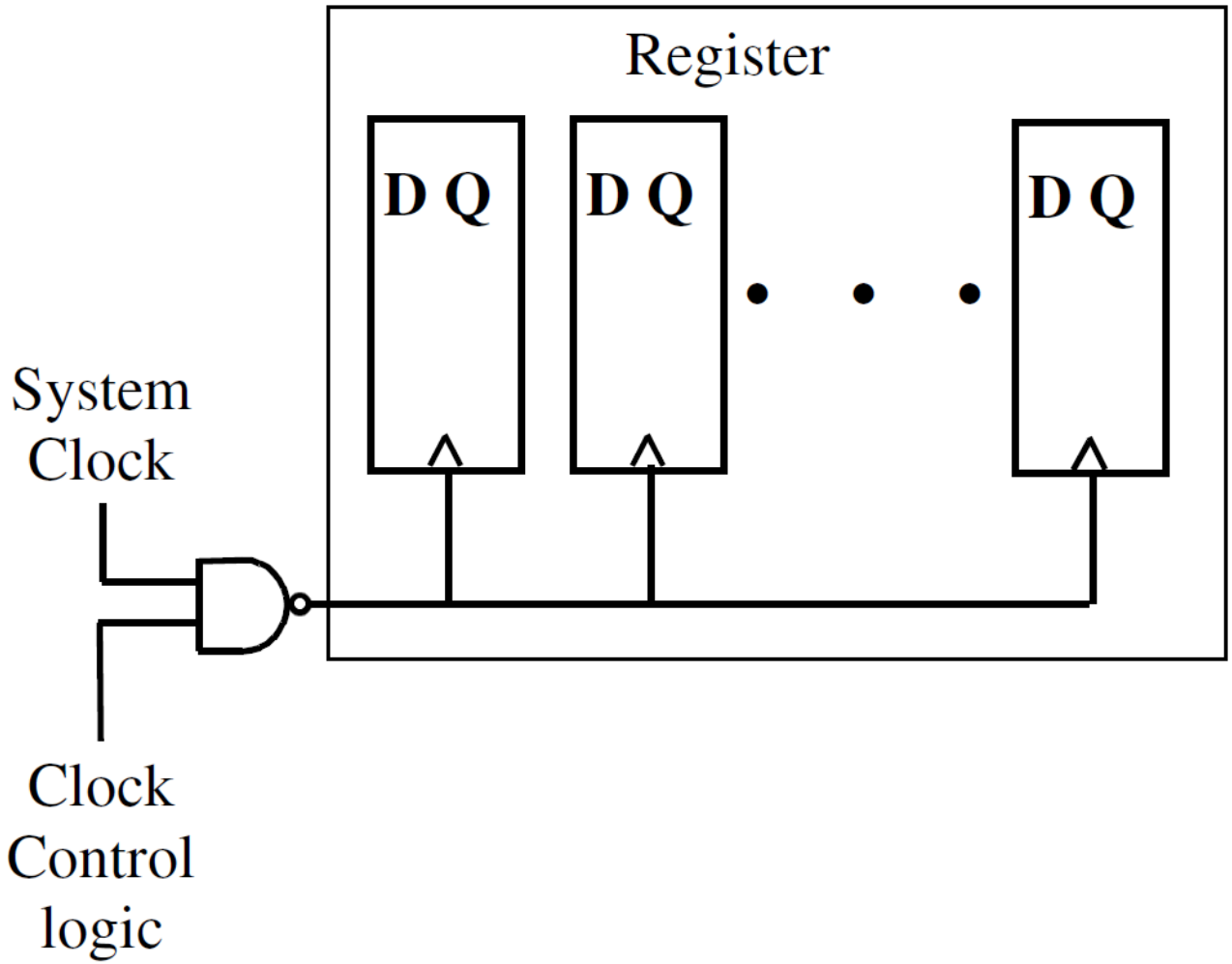
It consists of just two operations

MAR ← PC

IR ← RAM[MAR], PC ← PC+1

The second operation consists of two register transfers which occur in parallel.

Controlling the individual registers



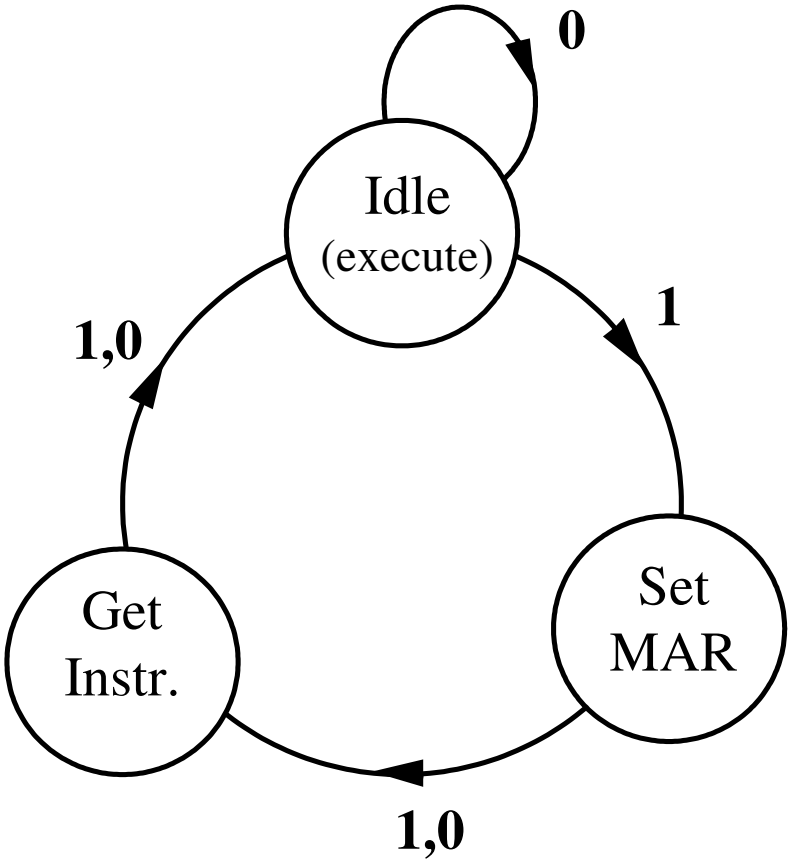
Register Transfers

The register transfers are achieved in two parts:

1. The multiplexers must be set to establish the required connections

2. The clock pulses must be connected to the registers which change their state.

The fetch cycle is a synchronous design problem



The *idle* state, will be replaced by an ‘execute’ cycle such as the one we used for the manual processor.

The fetch cycle output logic

	Clock Control			Multiplexer Control	
	MAR	**IR**	**PC**	**PC Input**	**MAR Input**
Idle	**-**	**-**	**-**	**-**	**-**
Set MAR	**1**	**0**	**0**	**X**	**0**
Get Data	**0**	**1**	**1**	**0**	**X**

Definitions

Zero sets the PC input multiplexer to select the incrementer

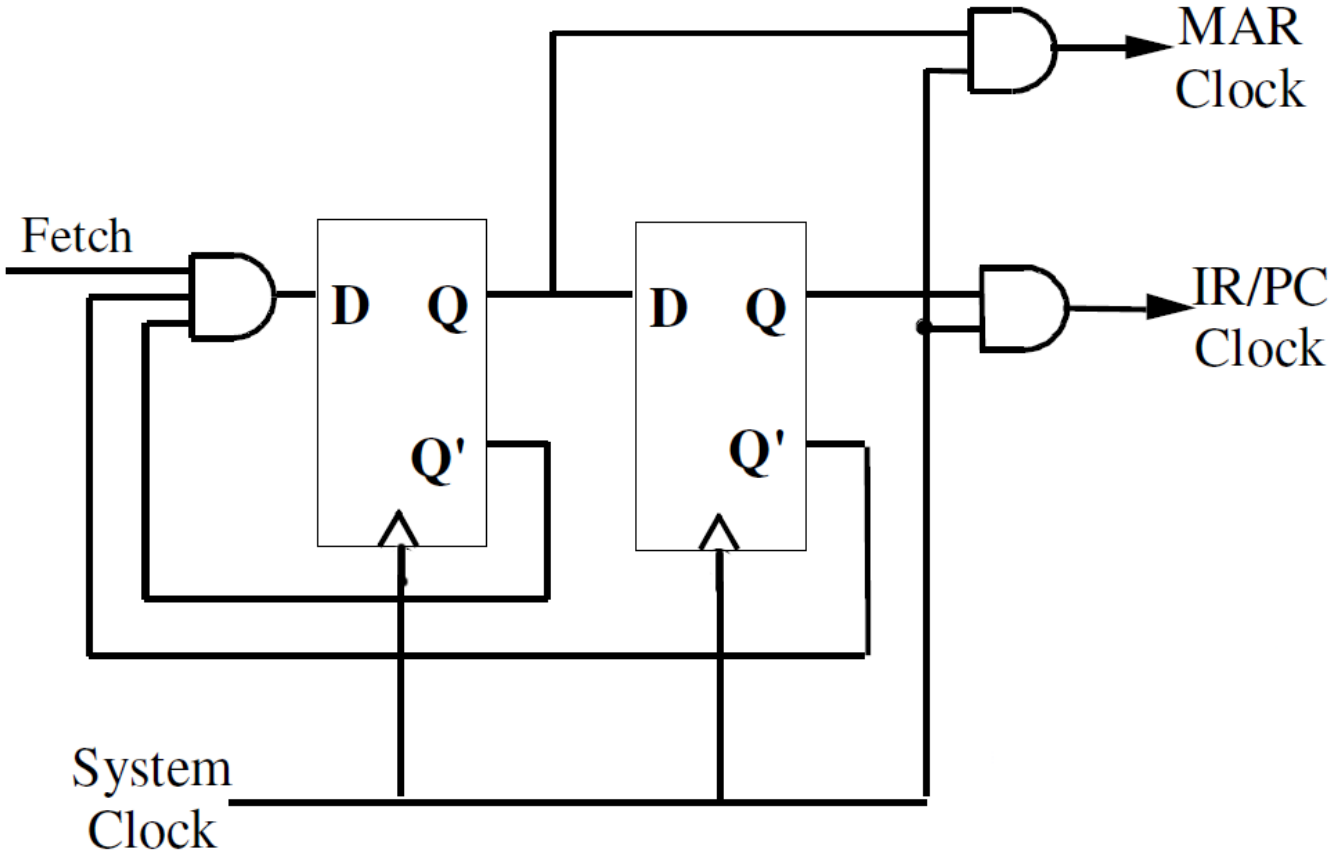
Zero sets the MAR input multiplexer to select the program counter

Using assignments

10 for the *set MAR* state

01 for the *Get Data* state

we can implement the output logic trivially


MAR
Clock
Fetch
D
Q
D
Q
IR/PC
Clock
Q'
Q'
System
Clock

Dynamic RAMs: DRAMs

For bulk RAMs (eg >1Mbit chips) D-Q flip flops are not used since they are too big.

Instead large RAMs use only one transistor and one capacitor for each bit.

Capacitor charged = 1 state

Capacitor uncharged = 0 state

Maintaining dynamic RAM

The store is not permanent and all the cells storing ones drift to zero in a fraction of a second.

So some method must be employed to restore the capacitor charge regularly

For this reason circuits of this kind are called dynamic RAM.

Refresh Logic

Refreshing the capacitor charges operates when the the computer is not accessing the memory.

A Memory Controller must tell the DRAM to refresh itself periodically….

This overhead is very low.