Memory Interleaving:

In this example we have 16 memory locations of 1 byte. Each memory location is addressed in binary from 0000 to 1111. To make memory access more efficient, this memory is spread out over four banks.

This system utilises high order interleaving where the first 2 bits of the address denote the bank and the last two denote the address withing the bank.

Memory Map

0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	I
1001	J
1010	K
1011	L
1100	M
1101	N
1110	O
1111	P

Module 00

00	A
01	B
10	C
11	D

Full address 0000

Module 01

00	E
01	F
10	G
11	H

Module 10

00	I
01	J
10	K
11	L

Module 11

00	M
01	N
10	O
11	P

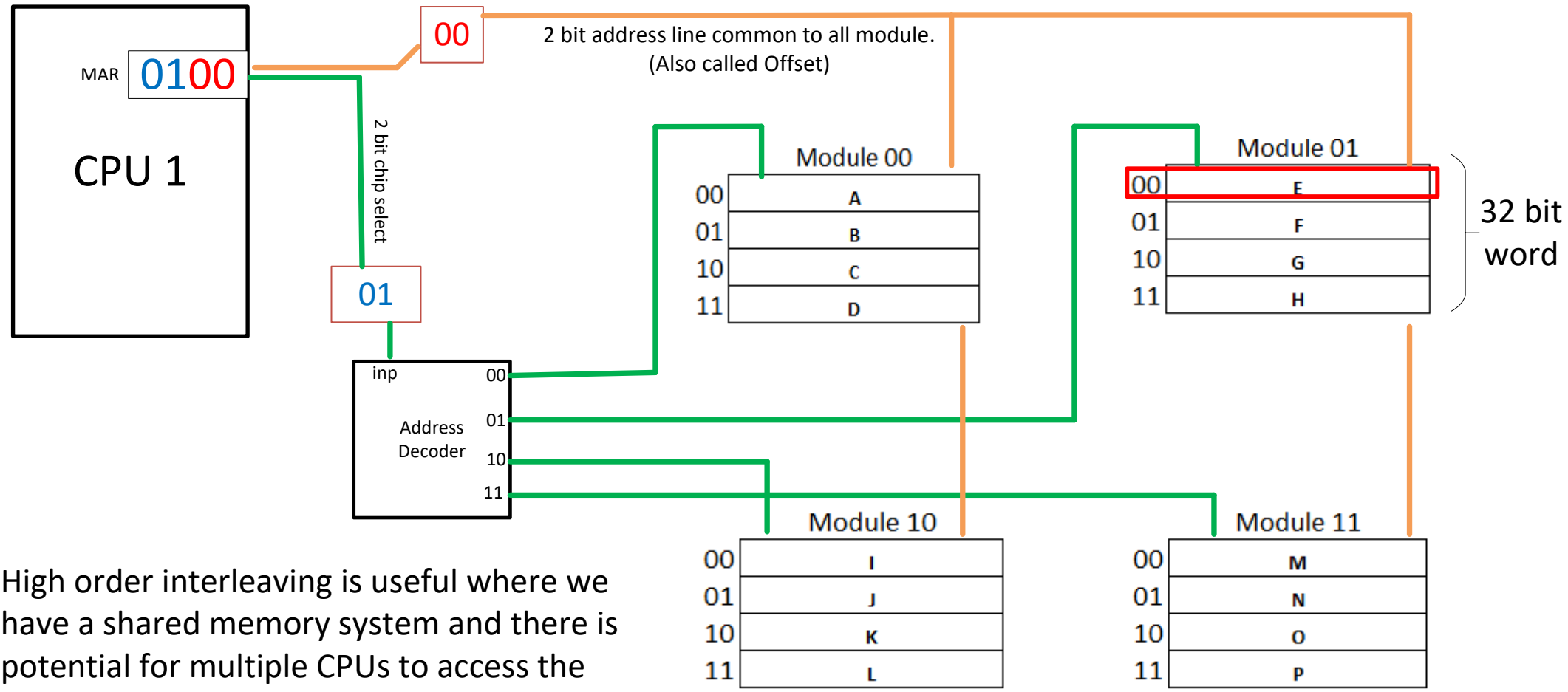
Full address 1111

The primary advantage of this interleaving memory across multiple banks is parallelism. It allows the memory controller to send data from each of the four banks simultaneously, as each bank operates independently of each other.

This example is high order interleaving, where the start of the address indicated which bank is being referenced, and the remainder are for address locations within the banks.

High Order interleaving is suitable for shared memory where multiple CPUs could be accessing banks independently.

High order interleaving: (Most significant bits select the module)



High order interleaving is useful where we have a shared memory system and there is potential for multiple CPUs to access the same memory pool.

In the example, CPU 1 is accessing Module01 and retrieving a 32bit word of data from 4 x 1 byte registers, we could have CPU2 accessing Module 10 at the same time.

A disadvantage of this design is if there is only 1 CPU, and it's accessing consecutive memory location on the same module, there is a time delay as all the work is being done on a single module, rather than being spread out over multiple modules.

Low order interleaving: (Least significant bits select the module)

This is an efficient design when we have a single CPU. Consecutive addresses are spread across the modules in order to load balance. If the CPU wanted to retrieve 4 words of data, the workload is across the 4 modules, rather than waiting for a single module to return all four words.