Basic Memory Operations

The memory unit supports two basic operations: *read* and *write*. The *read operation* reads previously stored data and the *write operation* stores a new value in memory. Both of these operations require a memory address. In addition, the write operation requires specification of the data to be written. The address and data of the memory unit are connected to the address and data buses of the system bus, respectively. The read and write signals come from the control bus. Two metrics are used to characterize memory. ***Access time*** refers to the amount of time required by the memory to retrieve the data at the addressed location. The other metric is the ***memory cycle time***, which refers to the minimum time between successive memory operations.

The read operation is nondestructive in the sense that one can read a location of the memory as many times as one wishes without destroying the contents of that location. The write operation, however, is destructive, as writing a value into a location destroys the old contents of that memory location.

Steps in a Typical Read Cycle:

1. Place the address of the location to be read on the address bus,
2. Activate the memory read control signal on the control bus,
3. Wait for the memory to retrieve the data from the addressed memory location and place them on the data bus,
4. Read the data from the data bus,
5. Drop the memory read control signal to terminate the read cycle.

Steps in a Typical Write Cycle:

1. Place the address of the location to be written on the address bus,
2. Place the data to be written on the data bus,
3. Activate the memory write control signal on the control bus,
4. Wait for the memory to store the data at the addressed location,
5. Drop the memory write signal to terminate the write cycle.

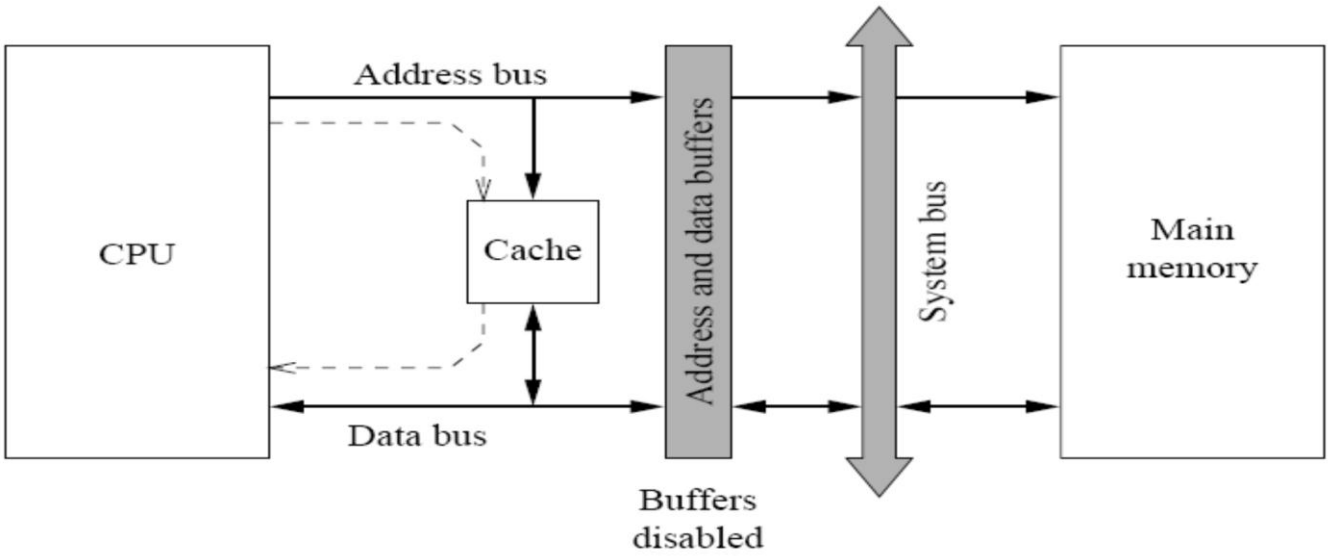
Cache Memory

Cache memory is much smaller than the main memory. Usually implemented using SRAMs, which sits between the processor and main memory in the memory hierarchy. The cache effectively isolates the processor from the slowness of the main memory, which is DRAM-based. The principle behind the cache memories is to prefetch the data from the main memory before the processor needs them. If we are successful in predicting the data that the processor needs in the near future, we can preload the cache تثبت and supply those data from the faster cache. It turns out that predicting the processor future access requirements is not difficult owing to a phenomenon known as *locality of reference* that programs exhibit.

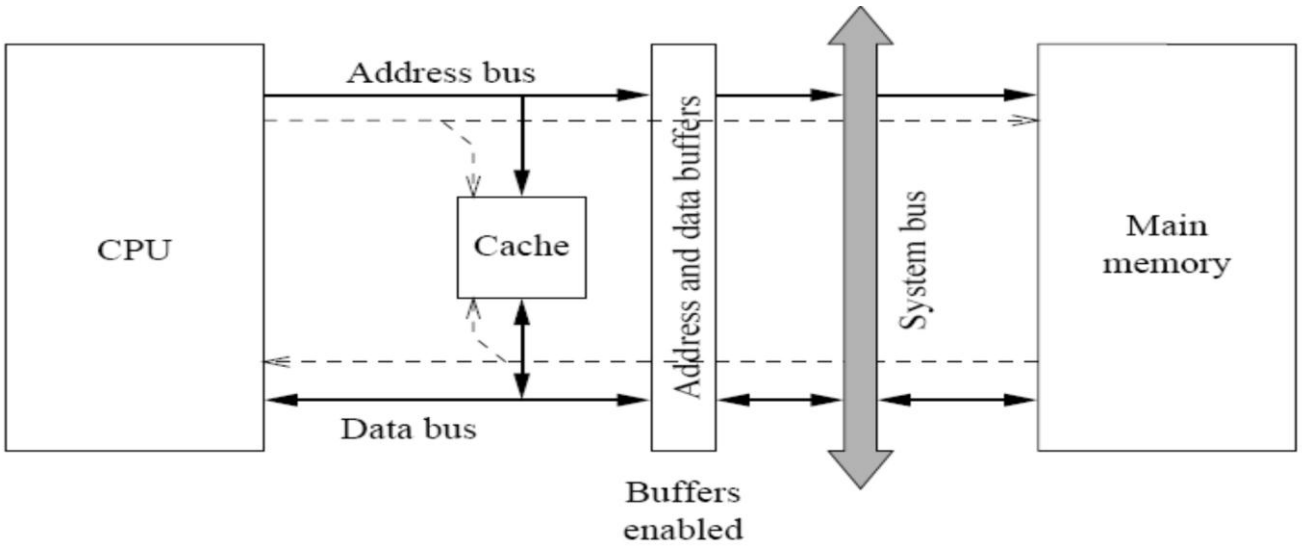
Performance of cache systems can be measured using the *hit rate* or *hit ratio*. When the processor makes a request for a memory reference, the request is first sought in the cache. If the request corresponds to an element that is currently residing in the cache, we call that a ***cache hit***. On the other hand, if the request corresponds to an element that is not currently in the cache, we call that a ***cache miss***. A cache hit ratio, hc, is defined as the probability of finding the requested element in the cache. A cache miss ratio (1 - hc) is defined as the probability of not finding the requested element in the cache. The likelihood of the processor finding what it wants in cache a high as 95 percent.

Figure (1) shows how memory read operations are handled to include the cache memory. When the data needed by the processor are in the cache memory ("read hit"), the requested data are supplied by the cache (see Figure 1a).

The dashed line indicates the flow of information in the case of a read hit. In the case of a read miss, we have to read the data from the main memory. While supplying the data from the main memory, a copy is also placed in the cache as indicated by the bottom dashed lines in Figure 1b. Reading data on a cache miss takes more time than reading directly from the main memory as we have to first test to see if the data are in the cache and also consider the overhead involved in copying the data into the cache. This additional time is referred to as the *miss penalty*.



(a) Read hit



(a) Read miss

In the case that the requested element is not found in the cache, then it has to be brought from a subsequent memory level in the memory hierarchy. Assuming that the element exists in the next memory level, that is, the main memory, then it has to be brought and placed in the cache. In expectation that the next requested element will be residing in the neighboring locality of the current requested element (spatial locality), then upon a cache miss what is actually brought to the main memory is a block of elements that contains the requested element (a ***block*** as a minimum unit of data transfer). The advantage of transferring a block from the main memory to the cache will be most visible if it could be possible to transfer such a block using one main memory access time. Such a possibility could be achieved by increasing the rate at which information can be transferred between the main memory and the cache. The transferring of data is referred to as a mapping process.

Modern memory systems may have several levels of cache, referred to as Level 1 (L1), Level 2 (L2), and even, in some cases, Level 3 (L3). In most instances the L1 cache is implemented right on the CPU chip. Both the Intel Pentium and the IBM-Motorola PowerPC G3 processors have 32 Kbytes of L1 cache on the CPU chip. Processors will often have two separate L1 caches, one for instructions (**I-cache**) and one for data (**D-cache**). A level 2 (L2) high speed memory cache store up to 512 kilobytes of data.

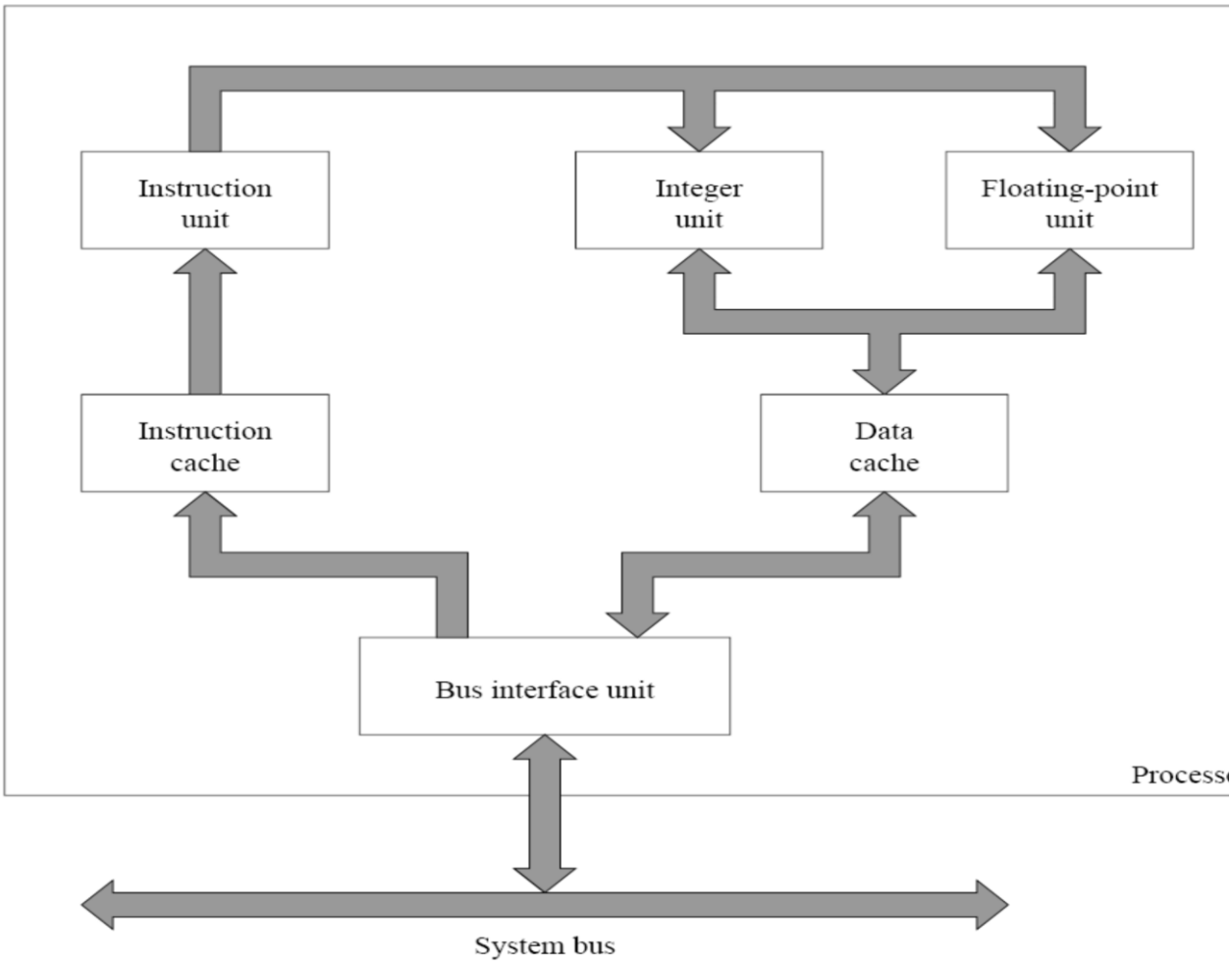


Figure 18 Most current processors use separate caches for instructions and data with separate instruction and data buses.