Paging

M. Sonza Reorda

Politecnico di Torino
Dipartimento di Automatica e Informatica

Introduction

The mechanism used by the 80386 to generate memory addresses when working in protected mode leads to 32-bit *linear addresses*.

Linear addresses must then be converted into physical addresses, taking into account the size of the available RAM memory, and the position in memory of segments.

This conversion is performed through the *paging* mechanism.

Pages

Mapping linear addresses to physical addresses is performed by making reference to units denoted as *pages*.

From the 80386, pages have a 4Kb size.

The total address space is divided in 2^{20} pages. Only some of them are in the main memory at a given time. The others could be in the secondary memory.

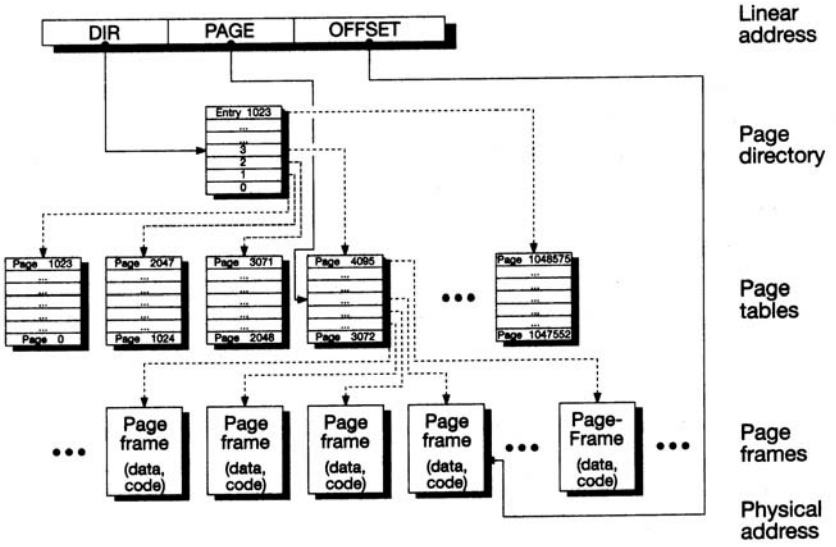
The PG bit in CR0 determines whether the processor must implement the paging mechanism or not.

Converting addresses

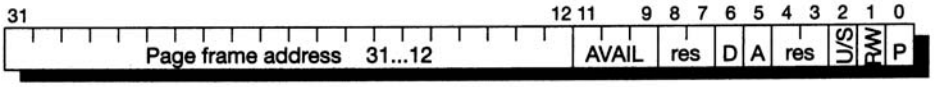
Conversion from linear addresses to physical ones happens using a two-level mechanism:

- **First, the 10 most significant bits in the address are used as an offset to access a *page directory* (pointed to by the CR3 register)**
- **Secondly, the following 10 bits are used to access one entry in the *page table* identified by the entry in the page directory; the former provides the page physical address**
- **Finally, the page physical address is combined with the *offset*, corresponding to the least significant 12 bits of the address.**

Address mapping



Page table entry



Page frame address: address bits 31...12 of the page frame
AVAIL: Available for the operating system

D:	Dirty	
	0=page has not yet been overwritten	1=page has been overwritten
A:	Accessed	
	0=page has not yet been accessed	1=page has already been accessed
U/S:	Page access level (User/Supervisor)	
	0=supervisor (CPL=0...2)	1=user (CPL=3)
R/W:	Write protection (Read/Write)	
	0=page is read only	1=page can be written to
P:	Page present	
	0=page is swapped	1=page resides in memory
res:	Reserved (equal 0)	

Page fault

If the P bit in the page accessed table entry is 0, the page is not in the main memory.

A *page fault* exception is triggered, and the OS is asked to move the page into the main memory.

In order to better chose the pages to be removed from the memory, the A bit can be exploited, telling which pages have been accessed in the last period.

TLB

To avoid performing memory accesses to support address translation, an on-chip cache is implemented, storing the most recently used page table entries.

This cache is called *Translation Lookaside Buffer* (TLB).

Each time an address translation has to be performed, the TLB is accessed first, and only if a miss happens, the access to the page tables arises.

The 80386 TLB is a four-ways set-associative cache, which uses a random replacement strategy.

TLB (II)

The TLB contains 4 sets, each including 8 lines. Each line is composed of a 24-bit tag field, and a 20-bit data field. The tag field stores the 18 most significant bits of the linear address (2 are implicit), a validity bit, and 3 attribute bits.

In this way the address conversion can normally happen very quickly, without accessing to the main memory.

Intel claims that a 98% hit rate can be reached by the TLB.

Page protection

Pages can be protected by specifying the privilege level required to tasks in order to access them.

Pages are marked as

- ***User*, if they can be accessed by any task**
- ***Supervisor*, if they can be accessed only by tasks having CPL<3.**

Dirty bit

Each page table entry stores a bit (D), which is set when the page is accessed for a write operation.

If D=1, when the page is to be removed from memory, its content must be copied to the secondary memory, to reflect changes arisen during its stay in the main memory.

R/W

Pages can be protected against any write operation by forcing to 0 the R/W bit in the corresponding page table entry.

Segmentation and paging