**The right-hand side of the micro-processor**

**THE MECHANISM ON 16 BITS**

* 6 registers on 16 bits and 1 on 32 bits

ADR (Architecture Decision Record) Device

1KB = 2^10 bytes

1MB = 2^20 bytes - we need 20 bits adresses

1 GB = 2^30 bytes

***How the adressing mechanism was designed for intel?***

* (the reason for which the addressing mechanism today has its root in 1979(?))
* When the people wanted to build a computer using intel with 1MB of RAM, the developers had to create a processing mechanism that could compute up with 20 bits addresses. **But how do you do that on 16 bits?** We need a logic mechanism to use more than one register. Therefore, you need a 2-step mechanism. They created a memory segment **(the „street”)**. This segment was named/initiated for every address and after that, you can specify how far from the beginning of the segment you go. Any beginning of the memory segment must also be a 20 bits address, so they proposed that **(RULE)** **any segment and every segment will start only at a multiple of 16 bytes. (RULE)**. The last 4 binary digit from the 20 needed are 0, so a 16 bit entity is enough to represent the most significant digits (the last 4 binary digits are also 0)
* Offset: the max of an offset is 2^16.
* In 16 bit programming the maximum size of a memory segment is 64 kb.
* THIS IS ONLY AN ADRESS SPECIFICATION. Who is counting the final address? We use a formula, an address computation: You only use the first 16 bits from the segment as a number and will multiply it by 16 and add the offset: **[segment\_code]\*16 + offset** – the REAL mode programming
* ONLY THE ADR COMPONENT CAN KNOW THIS FINAL NUMBER

ADRESS SPECIFICATION:

Segment : offset (how far from the beginning of the segment you go)

16 bits 16 bits

**Address** -> an address is associated to a memory location (there is no such thing to an address to a register or a bit, we talk only about RAM memory)

* An address is a number of the consecutive bits from the beginning of the ram memory and the beginning of the memory location that we are talking about

**\*COMPARISON TO REAL LIFE**

You are allowed to build 100 houses. They will not give you construction authorization, they will only give you a number (assign an address). Now, you have an address, you also need an addressing mechanism by which you access the address.

100 houses = 10^2 houses (00 -> 99) **(IMPOORTANT: IT USES TWO DECIMAL DIGITS)**

1000 houses = 10^3 (000 -> 999) **(IT IS COMPOSED OF THREE DIGITS IN BASE 10)**

THE 2 STEP MECHANISM: You don’t have one single number for every house, instead you have a street and you say how far from the beginning of the street you are. They used the same principle in creating the PC back in 1979

C and assembly – value oriented language

**THE ADRESSING MECHANISM ON 32 BITS**

The big difference:

1 Gb is 2^30

4 Gb is 2^32 bytes, we need 32 digits in base 2

If we would have been able to start this design 42 years ago directly with 32 bits registers we wouldn’t have had the necessity of performing 2 step mechanism. We would have already had the 32 bit register which would have been enough. From a logical point of view, a single structure would have been needed.

THE MEMORY FLAT MODEL: you don’t have memory segments with start and finish, only one long memory segment

You no longer have the starting address of memory segment. It is a very protective way of memory processing. It is hidden by the operating system which is hiding all the segments that are possible to be handed to the programmer. The operating system is managing a selector table

**SEECTOR TABLE**

|  |  |  |
| --- | --- | --- |
| SELECTOR | Base | limit |
| 5 | B0, B2, - b7 | L0 – l7 |
| 17 |  |  |
| 325 |  |  |
| 1801 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

IN 32 BITS PROGRAMMING, THE SEGMENT REGISTERS ARE ALSO ON 16 BITS (since it is enough), but you need to extend the offset part

At runtime, you see if that segment is something that is manageable and offered by the segment. The processor will ask: Do you have a selector with that value. If the operating system does not, an error occurs. If it does, it will offer to the processor the BASE (WHERE IT STARTS) and the LIMIT (HOW BIG IT CAN BE). These are informations that are hidden to programmers every time. This exchange is done only between the processor and the operating system. The processor checks if the offset you provided inside my segment in comparison to the limit. IF it is inside the limit, the address is valid, else an memory address violation error is offered. If both checks are processed the **adr** component are able to process the computations. The final adress : **adr = base + offset**

\*\*\*SUB 16 BITS STIU MEREU UNDE INCEPE UN SEGMENT, NU SI IN 32 BITS

THE SEGMENTATION MECHANISM : DEFINE THE START OF AN SEGMENT AND THEN DEFINE THE OFFSET

(16 bits) (toti 6 doar la programare in 32 bits, primii 4 in 16 bits) CS, DS, SS, ES, FS, GS – segment registers, they are identifying the segments which we are working with

FS, GS – DON’T HAVE SPECIAL ROLE, the others do

LOGICAL SEGMENTS

CS – Code segment(?)

DS – Data segment(?)

SS – Stack segment

ES – Extra segment

We can have thousand of segments that are completing a program. however, We can only have one segment of any one of these 4 types at a time. Which of the 4 are mandatory and which are mandatory (logical)?

IT IS MANDATORY FOR A STACK SEGMENT TO EXIST. IT IS NOT MANDATORY TO DEFINE IT

Mandatory

* Code segment

Optional

* data segment
* stack segment
* extra segment

How do you know which is extra segment? You use segment registers. These registers will contain only the segment selectors corresponding to every segment of every active segment type.

LET’S CHECK FOR 8:1000h

The processor will check if 8 is a selector accepted by the operating system. If not, the processor will say invalid memory adress, memory violation. If it exists, you get the base and limit. If 1000h is smaller than the limit and therefore the adress is valid.

EIP – EXTENDED INSTRUCTOR POINTER (ON 32 BITS)

* it is the combination of
* THE FIRST COLLUMN is the offset of every instruction inside the debugger

FORMULA DE LA 2 NOAPTEA

**Offset specification formula: [base] + [index x scale] + [constant]**

***(!2am formula)***

**DE CE [NUME] [DESTINATIE] [SURSA]?**

In 1932 an article stated that an electronic computer had to be composed of these compartments: center processor unit (cpu), a memory (volatile or remnant). 14 years later the first electronic computer was created based on this concept. Any architecture is a John (?) architecture

ANY PROGRAM IS A SEQUENCE OF ASSIGNMENT STATEMENTS