SCS Összegzés

• Ex: j et1 ;jump

jump de la etajul 1

1. **Laws and principles that control the performance of a computer:**
   - **Moor's law** – definition and comments on suitability:

"The number of transistors on integrated circuits doubles approximately every two years."
(18 months law): "The performance of a computer is doubled every 18 month" (1.5 year), as a result of more transistors and faster ones.
Moor's law suggests that increasing the number of transistors would lead to both large, fast CPUs but also very small, cheap CPUs.
Moor's law has an impact on the whole design of a computer architecture, from designing I/O devices, busses for the system, multiple processors on a single chip, networks and communication, and also suggests reducing the costs of components.
This increase has been exponential, but specialists from Intel have predicted a limitation to 16 nm technology. Similarly, we have seen as examples, the saturation of the clock frequencies, capacity of internal (DRAMs) and external memories.

   - **Amdahl's law** – definition and consequences

The performance gain that can be obtained by improving some portion of a computer can be calculated using Amdahl's law. Amdahl's law states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of time the faster mode can be used.

$$\text{Speedup} = \frac{Performance\ for\ the\ entire\ task\ using\ the\ enhancement\ when\ possible}{Performance\ for\ the\ entire\ task\ without\ the\ enhancement}$$

Increase of speed depends upon:

- The fraction of computation time the improved component can be used;
- The performance gained by the enhanced execution mode.

This law suggests the way in which effort/cost in improving some particular components (memory, processor) should be concentrated.

-

- **Locality principles** – definition and consequences

Time locality: “if a memory location is accessed than it has a high probability of being accessed in the near future”. Consequence: bring the newly accessed memory location closer to the processor for a better access time in case of a next access => justification of cache memories.

Space locality: if a memory location is accessed, then its neighbor locations have a high probability of being accessed in the near future”. Consequence: bring the location’s neighbors closer to the processor for a better access time in case of a next access => justification of cache memories; and

Transfer blocks of data instead of single locations; block transfer on DRAMs is much faster.

- **Parallel execution principle** – definition and forms of parallelism

Definition: “when the technology limits the speed increase, a further improvement may be obtained through parallel execution”.

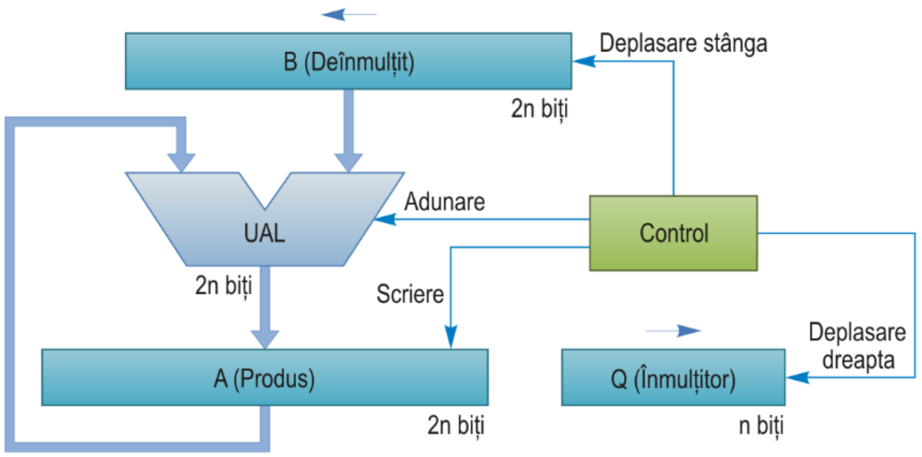
Parallel execution levels:

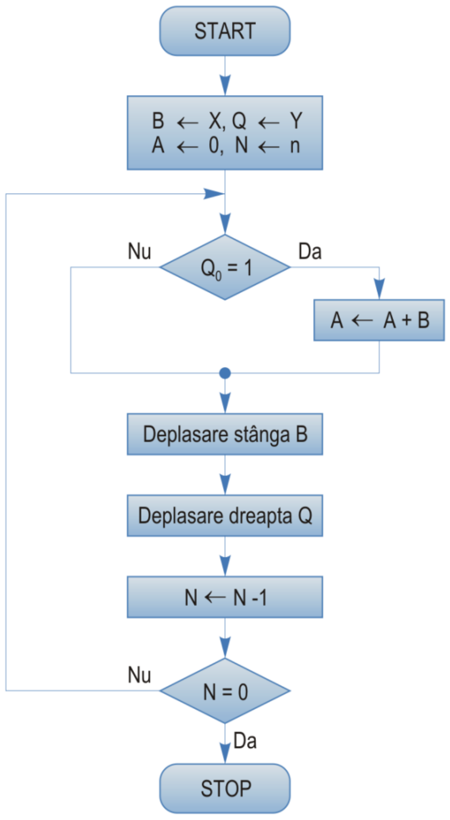
- Data level – multiple ALUs
- Instruction level – pipeline architectures, super-pipeline and superscalar, wide instruction set computers
- Thread level – multi-cores, multiprocessor systems
- Application level – distributed systems, GRID and Cloud systems.

Parallel execution is one of the explanations for the speedup of the latest processors.

1. Implementation of a multiply operation
    - Integer multiply (choose one):
        - Hardware scheme
        - The algorithm

Multiplication by shifting and addition:
- If the operands are n bit numbers, then the result will need 2n bits.





2013:

1. 6 performance features si 2 physical performance parameters

Answ:

- Performance features
    - Execution time
        - avg=Σ (tinstruction(i)*pinstruction(i))
    - Reaction time to external events
    - Memory capacity and speed
        - cache memory: SRAM, very high speed (<1ns), low capacity (1-8MB)
        - internal memory: SRAM or DRAM, average speed (15-70ns), medium capacity (1-8GB)
        - external memory (storage): HD, DVD, CD, Flash (1-10ms), very big capacity (0,5-12TB)
    - I/O facilities(interfaces)
    - Development facilities(programmers)
    - Dimension and shape
    - Predictability, safety and fault tolerance
    - costs: absolute and relative
- Physical performance parameters
    - Clock signal's frequency
        - clock period = delay(longest path)
        - = no_of_gates * delay_of_a_gate
        - clock period grows with the complex CPUs
    - Average instructions executed per second
        - $$\text{average\_no\_instr} = \frac{1}{\sum p_i * t_i}$$
    - Execution time of a program
    - number of transactions per second
    - communication bandwidth
    - context switch time

2. Multiplier (integer + floating point) Schema + Algorithm

Answ: at the top, **floating point ?**

3. Pipeline architectures. Hazards

Answ:

- Usual pipeline **CPI ~ 1**
- Superscalar : multiple pipelines, 2 instructions fetched in every clock cycle **CPI ~ 0.5**
- Superpipeline : phases require only half of a clock cycle **CPI ~ 0.5**

- **NO FREE MEAL!**
- Hazards:
    - Data hazard : Data dependency between consecutive instructions
        - RAW - present in all architectures, occurs often - solved with forwarding (common data bus)
        - WAR - It is rare in classic pipeline; more often in superscalar pipelines
        - WAW - pretty much the same as WAR
        - RAR - not a hazard
        - Data hazards can be solved by register renaming, forwarding, detection and stall phases
    - Control hazard : Jump/branch instructions change the normal (sequential) order of instruction execution
        - Solved by Stall phases, Branch prediction, Out-of-order execution
    - Structural hazard : Instructions in different phases use the same structural component (e.g. ALU, registers, memory, bus, etc.)
        - Solved by Detection and Stall phases, Redundant functional units, Harvard memory organization – separate code and data memory, Multiple buses, Out-of-order execution

4. Cache memory: Definition.Motivation.Implementation
5. Parallel architectures. Flynn. Limitations (Amdahl's law)

2021/2022:

Compute the avg. number of instruction executed per second by a processor which have the following types on instructions:

- Integer instructions executed in 5 clock periods, in 80% of the time
- Floating point instructions executed in 105 clock periods, in 20% of the time
- The duration of a clock cycle is 1 ns
- Express the value in MIPS

$$average\_no\_instr = \frac{1}{\sum p_i * t_i}$$

Int -> 5 clock periods, 80%
Float -> 105 clock periods, 20%

Avg = 1/(0.8 * 5 + 0.2 * 105) = 1/(4 + 21) = 1/25 = 0.04
Nem irtam oda az 5 melle hogy nanosec tehat 1/25^(10-9) => 40 MIPS

a. 450 MIPS
b. 40 MIPS
c. 100 MIPS
d. It can’t be competed

—--

Compute the normalized arithmetical weighted mean benchmark for a computer that executes the following set of benchmark programs in times specified in the following table:

$$B_{AM} = \frac{1}{n}\sum_{i=1}^{n} w_i * t_i$$

	Measured Power	Reference computer
Program 1	10	100
Program 2	100	10

Normalizaljuk a reference PC-re(magyarul elosztjuk 100-al az elso programot es 10-el a masodikat)

	Measured Power	Reference computer
Program 1	0.1	1
Program 2	10	1

Most a keplettel : (½)*(0.1+10) = 5.05

—--

How many adding operations are needed to perform the following multiply operation (no look-up table, simplest sequential method):

1100 1110 1111 0000 *

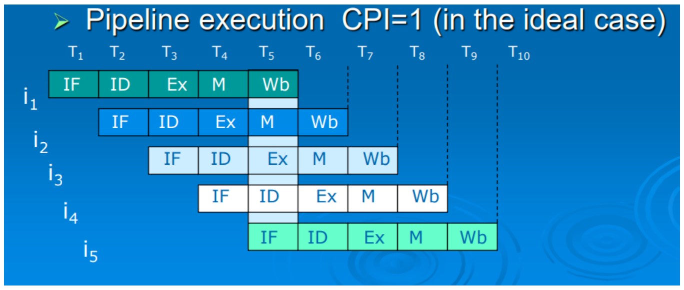
1010 1100 1100 1111

16 addition operations - the nr. of bits in the second number

SZABI SZERINT CSAK ANNYI AHANY EGYES, TEHAT 10

—--

The minimum (ideal) CPI (Cycles per instructions) for a classical pipeline CPU with 12 stages is:


Pipeline execution CPI=1 (in the ideal case)
T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9 T_{10}
i_1 i_2 i_3 i_4 i_5
IF ID Ex M Wb

a. 12
b. 6
c. 1
d. 1/12

Select the correct sentences:

a. Static branch prediction assures very good prediction for unconditional branch
b. Dynamic branch prediction assures very good prediction for conditional jump
c. Branch prediction is a way of solving data hazard cases
d. In case of branch prediction with saturating counters a change in the memorized prediction requires more than one miss-prediction
e. The miss-prediction rate does not depend on the type of application that is executed

---

Select the maximum limit for the speedup of a parallel computer system compared with a single processor system if the number of processors is 1000 and 80% of an application can be executed in parallel.

$$\frac{1}{1-q+q/n}$$

q = 80%, n=1000

1/(1-0.8 + 0.8/1000) = 1/(0.2 + 0.0008)) = 1/0.2008 ~ 4.98

a. 990
b. 5.55…
c. 1000
d. 4.98…

Select the proper access time for the following memory types

a. Around 1 clock period
b. Around 15 ns
c. 15-70 ns
d. 1-10 ms
e. Below 1 ns

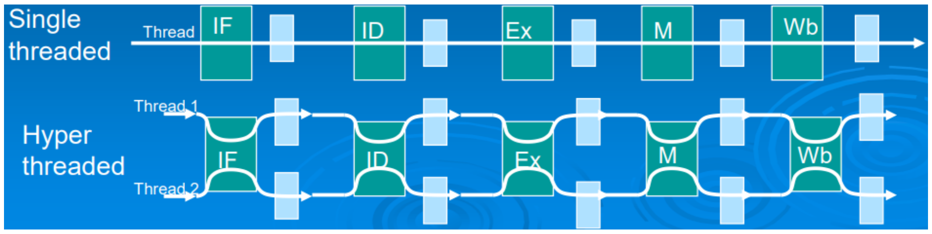
● Registers	[✓ a] b c d e
● Cache	a b c d [✓ e]
● HDD	a b c [✓ d] e
● SRAM	a [✓ b] c d e
● DDR DRAM	a b [✓ c] d e

Select the correct sentences related to segmentation:

a. Its main role is to protect the memory zones from unauthorized accesses
b. In protected mode, segments have variable length
c. Segments cannot be overlapped in the internal memory
d. Through segmentation the operating system can control the type of operations made in a segment
e. The selector of a segment contains flags that control the type of operations allowed in a segment

—---

Explain the following forms of parallelism:

- Data level parallelism (+ examples)
  - SIMD architectures (single input multiple data)
  - use of multiple parallel ALUs
  - it is efficient if the same operation must be performed on all the elements of a vector or matrix
  - signal processing, image processing, graphical rendering and simulation, scientific computations with vectors and matrices benefits from it
  - Example: Pentium II – MMX and SSE2
- Hyper threading
  - hyper-treading - parallel execution of instruction streams on a single CPU
  - Idea: when a tread is stalled because of some hazard cases another thread can be executed
  - two threads executed in parallel on the same pipelined CPU
  - after every stage two buffers (registers) store the partial results of the two threads
  - Speedup – approximately 30%
  - The operating system will detect 2 logical CPUs !!
  - 

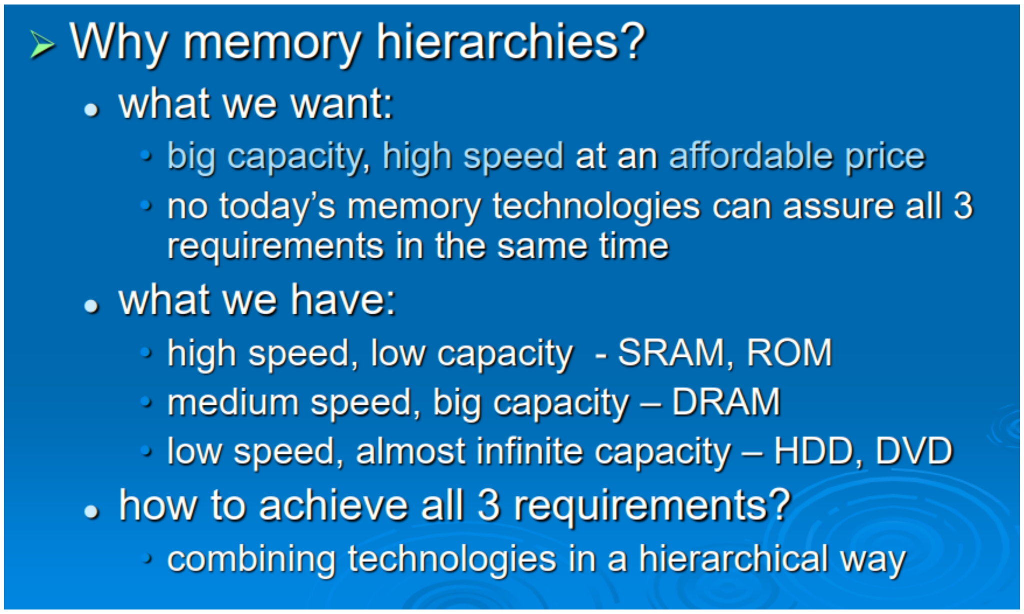
- Thread level parallelism (+ examples)
  - parallel execution at thread level
  - Examples:
    - hyper-threading – 2 threads on the same pipeline executed in parallel (up to 30% speedup)
    - multi-core architectures – multiple CPUs on a single chip
    - multiprocessor systems (parallel systems)
- Give 3 reasons for using parallel architectures
  - users want faster-and-faster computers
    - advanced multimedia processing
    - scientific computing: physics, info-biology (e.g. DNA analysis), medicine, chemistry, earth sciences)
    - implementation of heavy-load servers: multimedia provisioning
    - why not !!!!
  - performance improvement through clock frequency increase is no longer possible

        - power dissipation issues limit the clock signal’s frequency to 2-3GHz
    - continue to maintain the Moor’s Law regarding performance increase through parallelization

—---

Memory hierarchy:

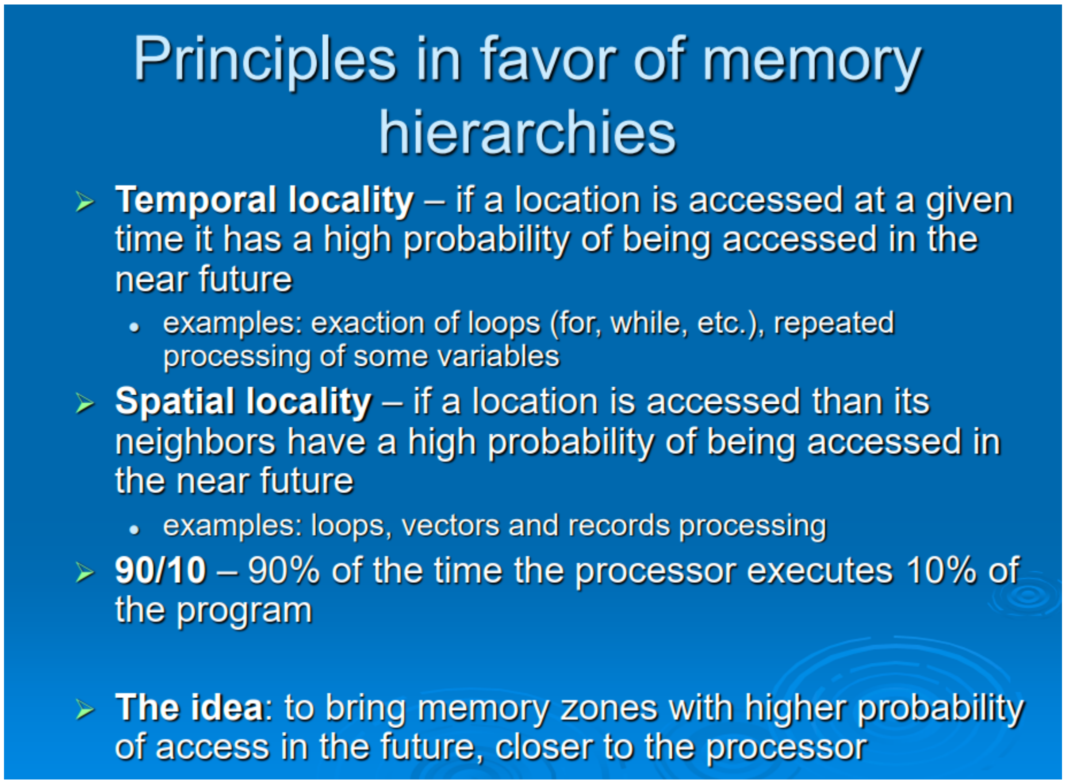
- Explain the need for such memory organization



Why memory hierarchies?

- what we want:
  - big capacity, high speed at an affordable price
  - no today's memory technologies can assure all 3 requirements in the same time
- what we have:
  - high speed, low capacity - SRAM, ROM
  - medium speed, big capacity – DRAM
  - low speed, almost infinite capacity – HDD, DVD
- how to achieve all 3 requirements?
  - combining technologies in a hierarchical way

- Express the principles in favor of memory hierarchy



Principles in favor of memory hierarchies

- **Temporal locality** – if a location is accessed at a given time it has a high probability of being accessed in the near future
  - examples: exaction of loops (for, while, etc.), repeated processing of some variables
- **Spatial locality** – if a location is accessed than its neighbors have a high probability of being accessed in the near future
  - examples: loops, vectors and records processing
- **90/10** – 90% of the time the processor executes 10% of the program
- **The idea**: to bring memory zones with higher probability of access in the future, closer to the processor

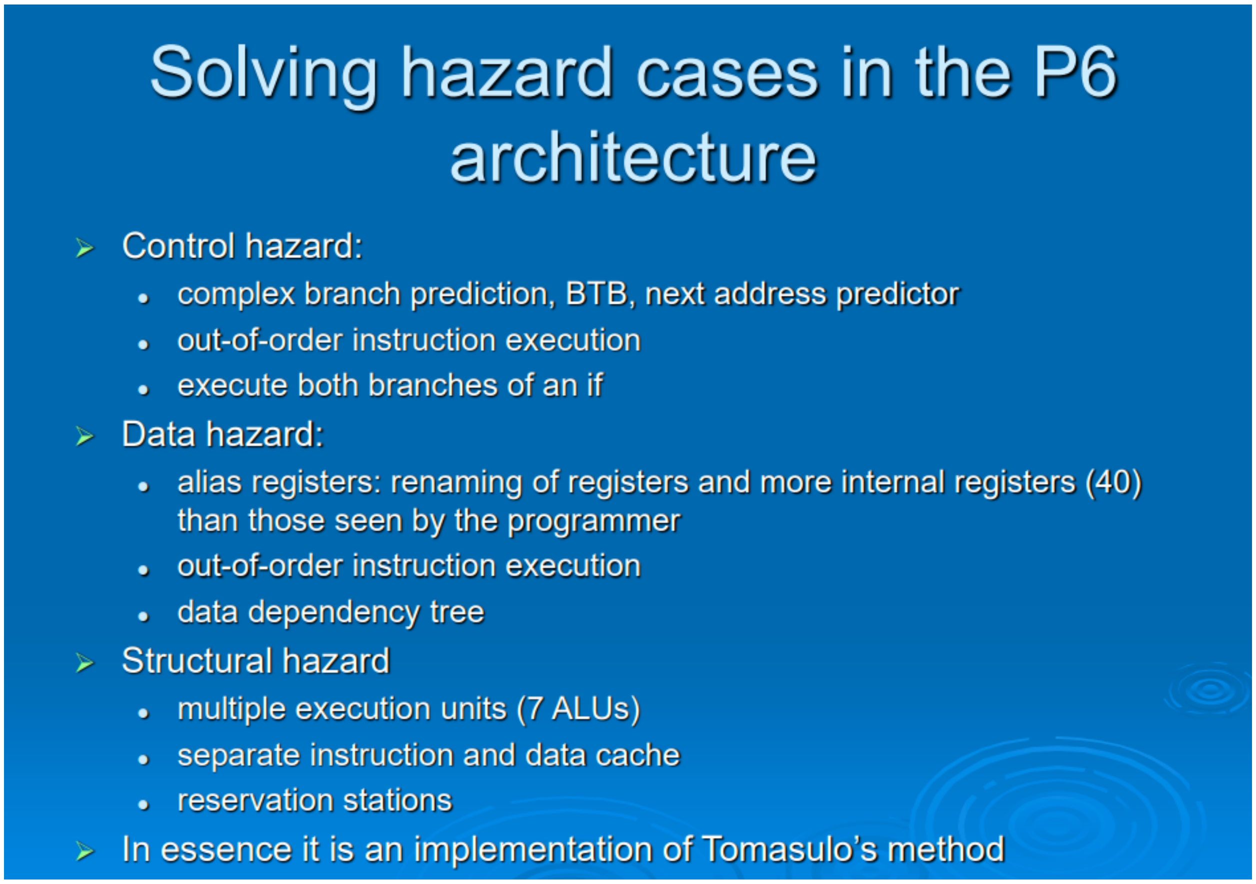
-

- Describe paging as a virtual memory technique
  - increase the internal memory over the external one
  - Internal and external memory is divided into blocks (pages) of fixed length
  - bring into the internal memories only those pages that have a high probability of being used in the near future
    - justified by the temporal and spatial locality and 90/10 principles
  - Implementation: similar with the cache memory – associative approach

—---

Related with the P6 architecture, explain the followings:

- Why is it called "superscalar architecture"
  - Superscalar because it can fetch and decode 3 instructions in parallel, and has 7 execution units (e.g. can execute 7 operations in one clock cycle)
- How does it differ from classical pipeline architecture
  - A classical pipeline architecture has only one pipeline per clock cycle, but a superscalar has multiple pipelines in a clock cycle => lower CPI is achievable
- How does the P6 architecture solve hazard cases



Solving hazard cases in the P6 architecture

- Control hazard:
  - complex branch prediction, BTB, next address predictor
  - out-of-order instruction execution
  - execute both branches of an if
- Data hazard:
  - alias registers: renaming of registers and more internal registers (40) than those seen by the programmer
  - out-of-order instruction execution
  - data dependency tree
- Structural hazard
  - multiple execution units (7 ALUs)
  - separate instruction and data cache
  - reservation stations
- In essence it is an implementation of Tomasulo's method

—---

Select the correct speedup of a computer with an improved graphical processing unit. This unit does graphical operations in 2 times faster than the previous version and the graphical processing represent 10% of the processing time of an application (Use Amdahl's Law)

$$\eta = 1 / [(1 - f) + f / \eta']$$

f = 10%, n'=2
n= 1/ (0.9 + (0.1)/2) = 1/(0.9 + 0.05) = 1/0.95 = 1.05

a. 2.01…
b. 1.05…
c. 1.95…
d. 0.5…

—--
Compute the normalized arithmetical mean benchmark for a computer that executes the following set of benchmark programs in times specified in the following table:

$$B_{AM} = \frac{1}{n}\sum_{i=1}^{n} w_i * t_i$$

	Measured power	Reference power
Program 1	10	100
Program 2	100	1

Normalizaljuk a referencehez(elosztjuk program1-et 100-al es program2-t 1-el)

	Measured power	Reference power
Program 1	0.1	1
Program 2	100	1

Most a keplettel : ½ (0.1+100) = 50.05
—--

The CPI of a superscalar architecture is:

a. Equal with 1
[✓ b. Smaller than 1]
c. Bigger than 2
d. It is unknown

—---

Select which kind of prediction is used in the case of the following instructions

a. Static prediction
b. Dynamic prediction
c. Saturating counters

- Procedure calls [✓ a] b c
- Unconditional calls [✓ a] b c
- Loops a b [✓ c]
- Conditional jumps a [✓ b] c

—---

Select the maximum limit for the speedup of a parallel computer system compared with a single processor system if the number of processors is 10 and 50% of an application can be executed in parallel

$$\frac{1}{1-q+q/n}$$ q=50% n=10

1/(0.5 + 0.05) = 1/0.55 = 1.(81)

[✓ a. 1.818]
b. 9.12
c. 10.5
d. 2.12

—---

Select the true sentences for a cache memory implemented with the associative technique

- It is a “1 way cache” memory
- [✓ A line from the internal memory can be placed in any position in the cache]
- [✓ It offers a better use of memory’s capacity compared with the direct mapping technique]
- [✓ The place of a line in the cache memory is obtained by comparing a part of the physical address with all the descriptors of the existing cache lines]
- [✓ A cache line is dropped from cache (transferred back to the internal memory) only when the cache memory is full and a new line is requested]

—---

Explain the principles of pipeline execution

- Explain hazard cases
    - Data hazard : Data dependency between consecutive instructions
        - RAW - present in all architectures, occurs often - solved with forwarding (common data bus)
        - WAR - It is rare in classic pipeline; more often in superscalar pipelines
        - WAW - pretty much the same as WAR

            - RAR - not a hazard
            - Data hazards can be solved by register renaming, forwarding, detection and stall phases
        - Control hazard : Jump/branch instructions change the normal (sequential) order of instruction execution
            - Solved by Stall phases, Branch prediction, Out-of-order execution
        - Structural hazard : Instructions in different phases use the same structural component (e.g. ALU, registers, memory, bus, etc.)
            - Solved by Detection and Stall phases, Redundant functional units, Harvard memory organization – separate code and data memory, Multiple buses, Out-of-order execution
- Give examples of methods used to reduces the negative effects of hazard cases
    - Introduced them before

—--

Define virtual memory

- extension of the internal memory on the external memory
- mechanisms for protecting memory zones allocated for different purposes

- Explain in detail the implementation of virtual memory through segmentation
    - Objective - divide and protect the memory zones from unauthorized accesses
    - Principles - how to implement
        - divide the memory into blocks (segments) that are fixed for variable length and can be with ot without overlapping
        - Address locations with **Physical_address = Segment_address + Offset_address**
        - Attach attributes to a segment in order to control the operations allowed in the segment and describe its content

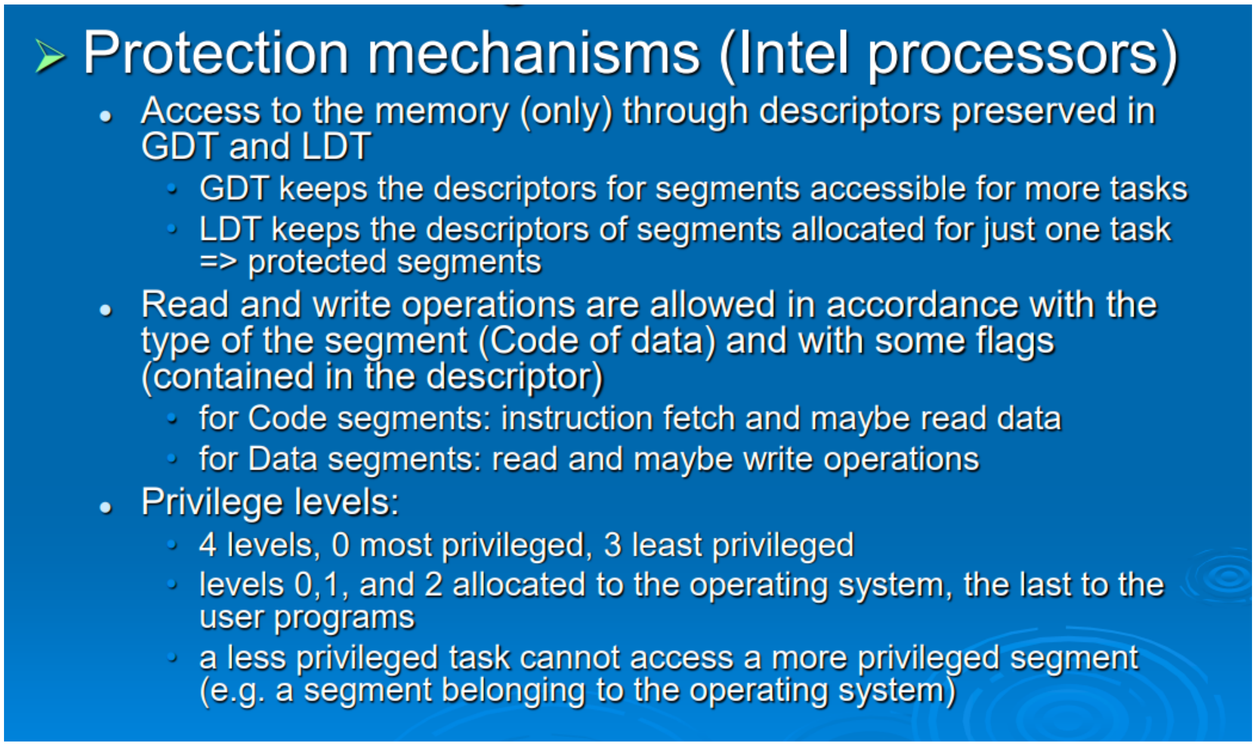
In a segmented virtual memory system, the computer's memory is divided into segments or blocks, each of which can be assigned to a specific program or process. Each segment has its own base address and limit, which define the range of memory addresses that the segment can access.

When a program is loaded into memory, the operating system assigns it one or more segments. The program can then access the memory within its assigned segments by using relative addresses, rather than physical addresses. This allows the program to access more memory than is physically available on the computer, since the operating system can move segments in and out of memory as needed.

To access memory outside of its assigned segments, a program must request permission from the operating system. The operating system can grant or deny this request based on the program's privilege level, and will swap segments in and out of memory as needed to make room for the requested segment.

When a program writes to a segment, the operating system checks to see if the segment is in main memory. If it is not, the operating system will swap the requested segment with a segment that is currently in memory but not being used.

- Present protection methods promoted through segmentation
    - In protected mode segmentation can have variable length



Protection mechanisms (Intel processors)

- Access to the memory (only) through descriptors preserved in GDT and LDT
    - GDT keeps the descriptors for segments accessible for more tasks
    - LDT keeps the descriptors of segments allocated for just one task => protected segments
- Read and write operations are allowed in accordance with the type of the segment (Code of data) and with some flags (contained in the descriptor)
    - for Code segments: instruction fetch and maybe read data
    - for Data segments: read and maybe write operations
- Privilege levels:
    - 4 levels, 0 most privileged, 3 least privileged
    - levels 0,1, and 2 allocated to the operating system, the last to the user programs
    - a less privileged task cannot access a more privileged segment (e.g. a segment belonging to the operating system)

—---

Make a comparison between a general purpose parallel asynchronous bus and a general purpose parallel synchronous bus

- Explain the meaning of parallel and general bus
    - parallel bus – transfer is made on multiple parallel lines (signals)
    - connect different components of a computer system (CPU, memory, interfaces, peripheral devices) - buses
- Explain the differences between the two buses
    - asynchronous – the bus in not controlled by clock signal; signals travel on the bus with a limited speed causing delays
        - single master – only one module (the CPU) can initiate transfers on the bus
        - multi-master – multiple modules can initiate transfers on the bus
    - Synchronous- every signal on the bus is related (synchronized) with the clock signal
        - modules may anticipate next steps (does not have to wait until a signal arrives to the module, as in asynchronous mode)
        - modules on the bus must have some intelligence

- Specify advantages and drawbacks for the two buses
    - Asynchronous :
        - simple operation (easy to understand and debug)
        - simple design of bus modules
        - no dimensional limitations (asynchronous mode)

        - single communication environment for all the components of a computer
        - low speed – limited to the slowest module
        - limited number of modules connected on the bus (10- 16 – see fan-out of a TTL circuit)
    - Synchronous:
        - higher transfer speed
        - small average access time
        - promotes block transfers (good for cache line transfers)
        - dimension of the bus is limited by the clock frequency
            - if the bus is too long, clock signal is not synchronized with itself at the two ends of the bus (the speed of the signal is limited)
        - more complex design of modules connected on the bus
        - harder debugging process
- Give examples for each case
    - Asynchronous:
        - 8086 bus
        - ISA (Industry Standard Architecture), EISA (extended ISA)
        - S-100
    - Synchronous
        - PCI
        - P6 (Pentium Pro) bus