Repetition

2. Boolean Logic, Circuits
5. Internal Memory.
6. External Memory.
8. CPU-Instruction Sets, Addressing Modes.
9. CPU Structure.
Importance

A = very relevant, B = small questions, C = irrellevant for the exam.

1. Historic development of computers. C
2. Boolean Logic, Circuits A
3. Computer Arithmetic and Representation of Data. A
5. Internal Memory. A
6. External Memory. C
7. Input/Output, Interrupts. B
8. CPU-Instruction Sets, Addressing Modes. A
9. CPU Structure. A (related to 8).

No guarantee for completeness.
Four Main Topics

- **Boolean Logic, Circuits**
  - Circuits, Flip flops, Circuits for addition.

- **Computer Arithmetic**
  - Binary representation, multiplication algorithms.
  - Signed numbers, floating point numbers, ASCII.

- **Internal Memory**
  - Cache: direct/associative/set-associative.
  - tag, line/set, word
  - replacement algorithms.

- **CPU instruction sets.**
  - Addressing Modes, assembler programs, instruction cycle execution.

Additionally small questions from (B- and A-) topics.
Best Preparation

(a) Coursework 1
(b) Coursework 2
(c) Sample exam 1.
(d) Sample exam 2.
(e) *Learn a few details.*
   (Notions; why does one do what?)
(f) *Consult old exams.*
Because: Most Question demand to *do something yourself.*
(Although only things you have done before).
2. Boolean Logic, Circuits

- Gates, Truth Tables.
- Circuits for defining arbitrary boolean functions. (How to really do it).
- Flip-flop (configuration, states, modification, memory storage).
- Adders (half, full, n-bit).
  (Fits as question better here than in 2.)
Trick with constructing circuits for boolean functions:

- Take inputs A, B, C, ....

- For every one in the result, take one AND gate and connect it with each of A, B, C, ....

- If A is 0 in the corresponding row, put a 0 at the gate.
- If B is 0 in the corresponding row, put a 0 at the gate.
- etc.
- OR everything.
### Example

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

![Diagram of logic gate circuit with inputs A, B, C and outputs (1, 0, 0) and (0, 0, 1)]
3. Computer Arithmetic

- Binary representation of numbers.
- Conversion boolean ↔ hexadecimal ↔ decimal.
- Subtraction. (Take bitwise complement and add one).
- Multiplication
  (4 algorithms with varying number of bits in sum).
- Signed numbers (twos complement). Conversion (Complement + add one).
- Signed fixed point numbers.
- Size of an ASCII file. Character codes.

- PC, IR, MAR, MBR.
- Execution of a small program.
- Instruction Cycle State Diagram (Instruction address calculation → instruction fetch → instruction decoding → ...).
- Buses: data/address/control lines. (Purpose).
- Multiple bus hierarchies (why, how?)
- Fixed, multiplexed, dedicated lines (what’s that?)
- Arbitration (centralized, distributed).
- Synchronous, asynchronous timing.
- Transfer types (read, write, read-modify-write, read-after-write, block transfer).
5. Internal Memory

- Byte/Kbyte/Mbyte etc.; bit
- Sequential/direct/random/associative access.
- Semi-conductor/magnetic/optical/magneto-optical.
- Volatile/non-volatile.
- Erasable/non-erasable.
- SRAM/DRAM
- Notions: RAM/ROM/PROM/EPROM/EEPROM/Flash;
  Read only/read mostly.
- Addressable up to byte level/word level.
  (what does this mean?).
Cache

- Why?
- Blocks stored in it (in order to reduce number of tags).
- Hit/miss.
- Write through/write back.
- Two level caches
- Split vs. unified cache.
- Direct mapped/associative/set associative cache:
  - Tag – line/set – word.
  - Four replacement algorithms.
More on Cache

Assume Block size is 16 bytes. Then in decimal representation:

- Byte 0 - 15 go to first block.
- Byte 16 - 31 to second block
- Byte 32 - 47 to third block
- etc.

If we have 4 lines and direct mapping then

- Block 0,1,2,3 go to line 0,1,2,3 respectively.
  (Block 0 is byte 0 - 15; block 1 is byte 16 - 31 etc. if blocksize = 16 byte).
- Block 4,5,6,7 go to line 0,1,2,3 respectively.
- Block 8,9,10,11 go to line 0,1,2,3 respectively.
- etc.
If we have 4 sets with 2 lines each then:

- Block 0,1,2,3 go to set 0,1,2,3 respectively.
- Block 4,5,6,7 go to set 0,1,2,3 respectively.
- Block 8,9,10,11 go to set 0,1,2,3 respectively.
- etc.

If addressable unit is word, count in words instead of bytes.
7. I/O, Interrupts

- Groups of I/O devices (input; output; input and output; storage).
- Three control methods: programmed I/O, interrupt driven I/O, DMA.
  How do they work (roughly).
- What is software polling, daisy chain?
- Memory mapped vs. isolated I/O?
- Parallel vs. serial ports.
8. CPU Instruction Sets

- Types
  - arithmetic (which?)/logic (which?)
  - Data transfer
  - Data I/O
  - Control (jumps, branches, skip, subroutine, stop, wait, Nop).
- Operator, operands
- Number of addresses.
- Reduction of number of addresses.
  (Examples, how to compute a simple term!)
- Addressing modes
  (Which operand is loaded in “Load register indirect 3”, if R3 = 10, memory location 10 has content 20?)
  (No fancy addressing modes, ie. vi.3 - 5, Pentium II addressing modes).
- Example of assembler computing with an array.
- Instruction formats. What parts, how to build it.
- How does stack, subroutine jump work.
9. CPU Structure

- Three parts: ALU, CU, registers. Tasks, basic behaviour.
- User-visible/control registers (what kind).
- Control lines, data lines.
- Basic flow of data (don’t learn slide 283 by heart).
  (How do data flow in an instruction cycle if I have PC, address bus, data bus, IR, and an adder for calculating PC + 1
  Possibly with MAR, MBR instead – just replace address bus by MAR and databus by MBR.
- Hard-wired control vs. micro-programming (why); firmware.