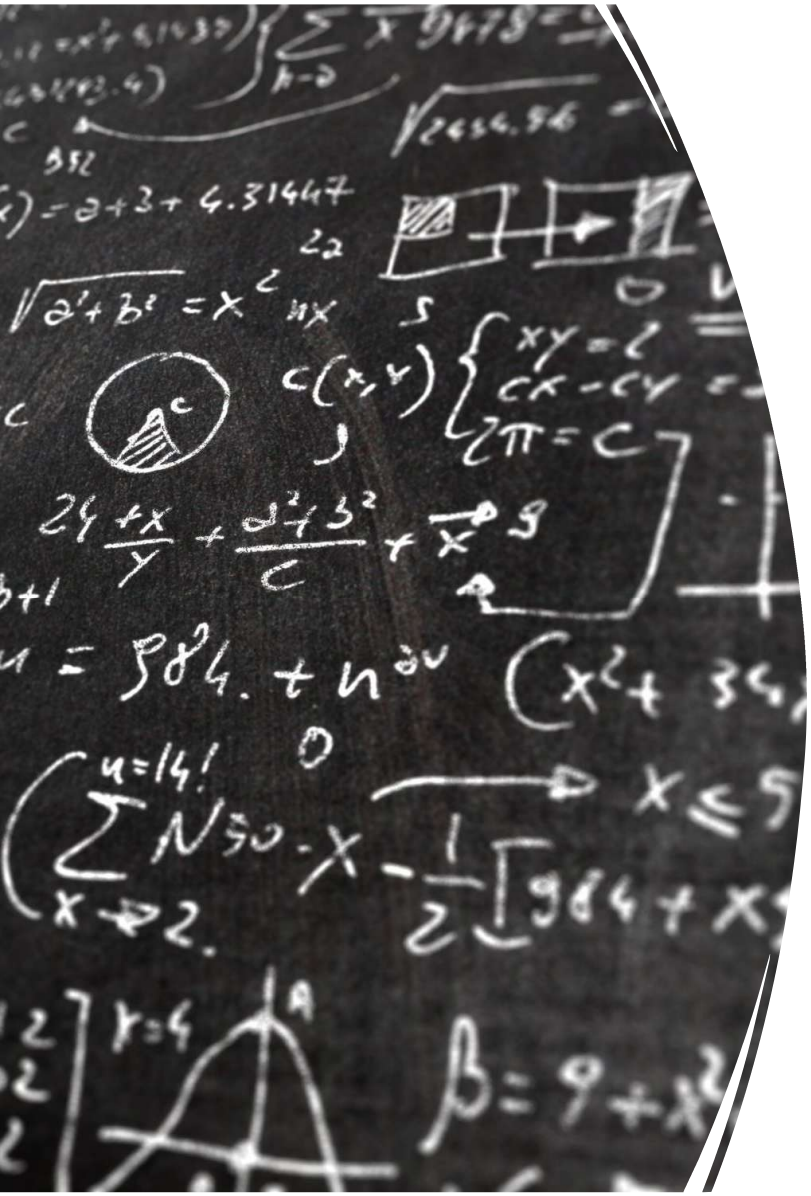
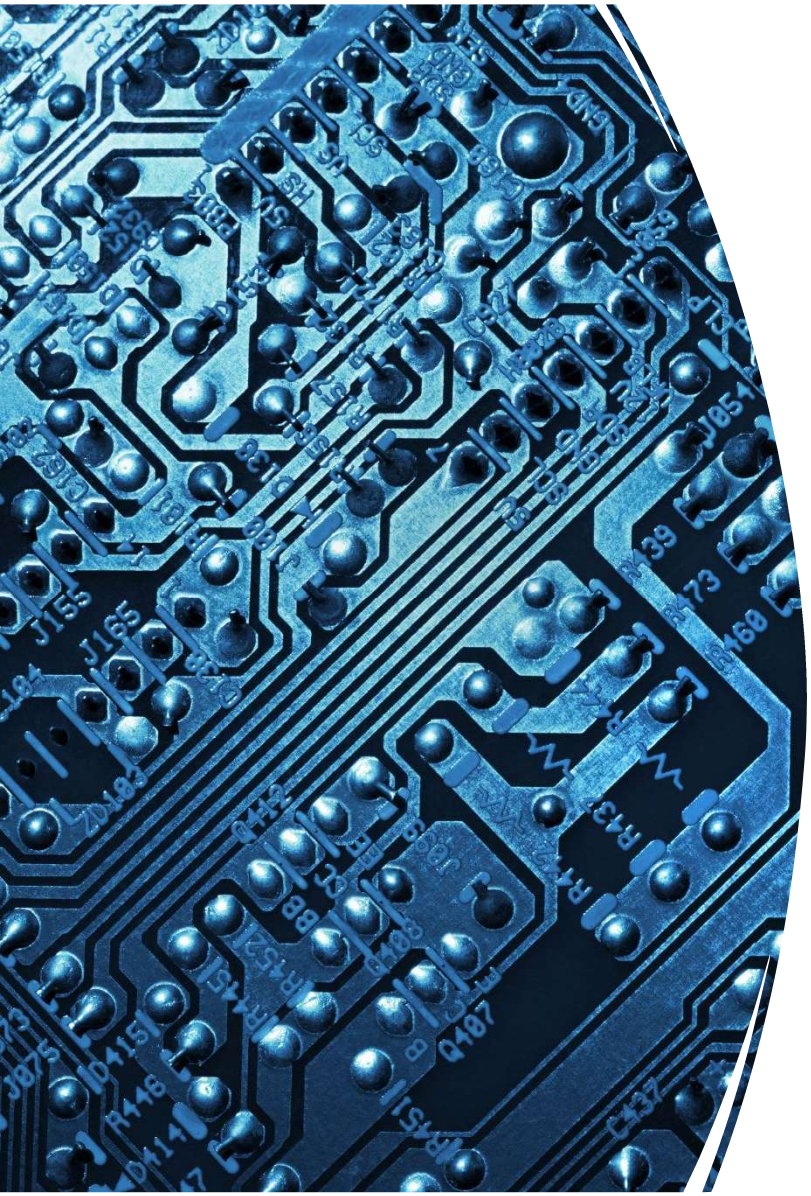


Instruction Memory

MIPS32 Single Cycle Processor



Function

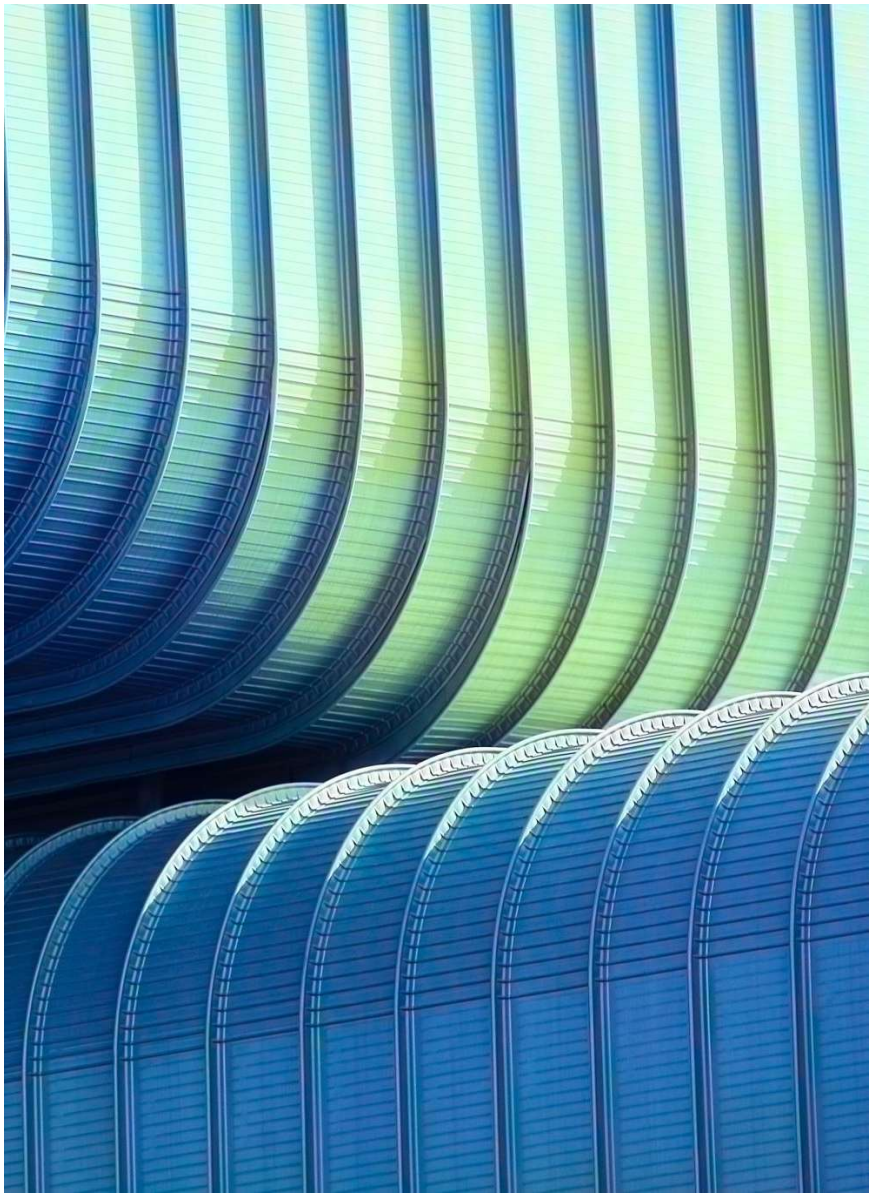
- It stores the program instructions (in binary machine code) and provides the current instruction to the processor based on the program counter (PC) value
- Each assembly instruction maps directly to an instruction in memory
- Instructions come from when code in a high-level-language like C or C++ is compiled into machine or assembly
- The memory output is when the opcode of the instruction is passed to the control rom to provide signals to the other processor components like data memory and the register file

Importance

The instructions have to be in memory to tell the processor what to execute

The instruction memory stores all the instructions for a program

The instruction is decoded in the control rom to produce signals for the operation of the ALU, register file, data memory, and selectors for the multiplexers



Difficulties

Machine code instructions must be properly read in from a file

Output must be properly fed into multiplexers, registers, and data-memory or else incorrect output will be produced, and instructions will not execute properly

Pros and Cons

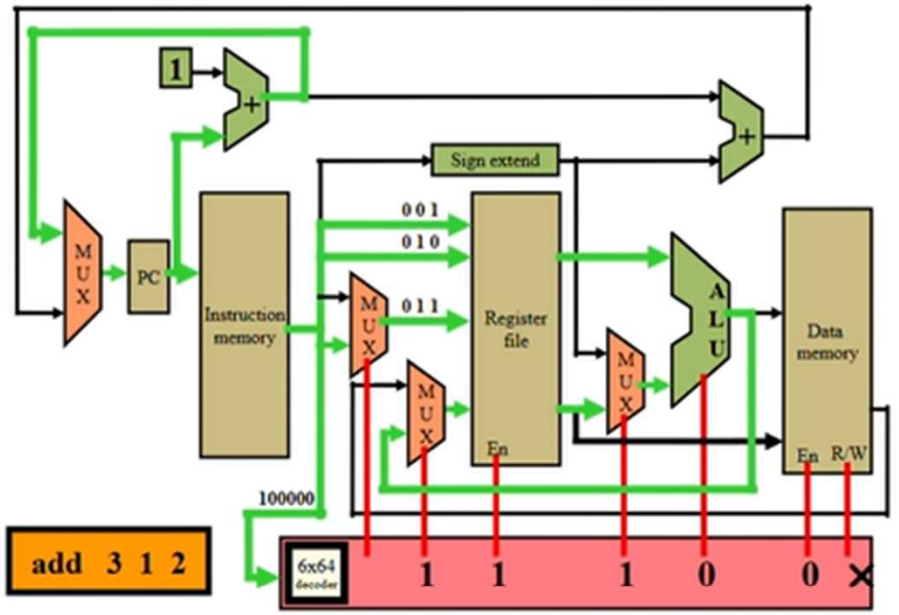
Pros

- Holds the actual instructions that the program must execute
- Without it no computation or control logic could be performed
- Implemented as rom
- Allows quick instruction fetch, especially important in signle-cycle and pipelined CPUs
- Improves performance by allowing instruction fetch and data access in parallel
- Fixed-size instruction words(e.g., 32 bit in MIPS) simplify addressing and fetching logic

Cons

- Maximum addressable memory is capped at 4GB(2^32 bytes)
- Simple decoding but code becomes large binary files since each instruction is 32 bits

Diagram


1
+
+
Sign extend
M
U
X
PC
Instruction memory
001
010
011
M
U
X
Register file
En
M
U
X
M
U
X
A
L
U
Data memory
En
R/W
100000
add 3 1 2
6x64
decoder
1
1
1
0
0
X