布林值： 表示兩種可能的狀態真或假，以及 0 或 1。

NAND：

NAND 邏輯閘是一種通用的邏輯閘，這意味著任何布林函數都可以僅使用 NAND 閘來實現。

NOT(x) = x NAND x4

邏輯閘：

基本邏輯閘： NOT、AND、OR、XOR5678

多位元邏輯閘： NOT16、AND16、OR16、MUX16

組合邏輯閘： 由基本邏輯閘組合而成的更複雜的邏輯閘

可以使用邏輯圖來表示組合邏輯閘的結構。

HDL： 一種用於描述硬體電路的語言.用於描述邏輯閘的介面和實現。

邏輯閘： 數位電路的基礎建構單元，以布林邏輯為運算基礎。常見的邏輯閘包括：

AND：兩個輸入皆為 1 時，輸出為 1，否則輸出為 0。

OR：兩個輸入中至少有一個為 1 時，輸出為 1，否則輸出為 0。

NOT：反轉輸入的邏輯值，0 變為 1，1 變為 0。

XOR：兩個輸入不同時，輸出為 1，否則輸出為 0。

二進制數是一種基於 2 的位置記數系統，每個位置代表 2 的冪次方。

計算機中的電路只能表示兩種狀態（0 或 1）

布林算術用於對二進制數執行算術運算，例如加法和減法。

二進制加法遵循與十進制加法相同的規則，但進位發生在逢二進一位時。

計算機使用二的補數來表示帶符號整數。

在二的補數中，最高有效位元代表符號位元，0 表示正數，1 表示負數。

將一個數字轉換成其相反數，只需要將所有位元反轉並加 1。

使用二的補數，加法和減法可以使用相同的演算法來完成。

溢位 (overflow) 發生在運算結果超出可表示範圍時。

半加法器 (Half Adder): 最簡單的加法電路，接受兩個一位元輸入，產生兩個輸出：和 (sum) 與進位 (carry)。

和：使用 XOR 閘實現，兩個輸入不同時輸出為 1。

進位：使用 AND 閘實現，兩個輸入皆為 1 時輸出為 1。

全加法器 (Full Adder): 考慮到進位的加法電路，接受三個一位元輸入（兩個加數和一個進位），產生兩個輸出：和與進位。

全加法器可以由兩個半加法器和一個 OR 閘組合而成。

多位元加法器： 將多個一位元加法器級聯，即可建構出多位元加法器，實現更複雜的算術運算。

例如，一個 16 位元的加法器可以由 16 個全加法器級聯而成。

鏈波進位加法器 (ripple-carry adder) 以及速度更快的 前瞻進位加法器 (carry-lookahead adder)。

減法器： 可以利用加法器實現，將減數取二補數後與被減數相加即可。

ALU 是一個 執行算術和邏輯運算的組合電路

Hack ALU 是一個運算兩個 16 位元二的補數值的 ALU，並輸出一個 16 位元

二的補數值。

Hack ALU 具有 18 個輸入和 3 個輸出。

輸入包括兩個 16 位元的運算元 x 和 y，以及 6 個控制位元，用於選擇要執行的運算。

輸出包括一個 16 位元的結果 out，以及兩個狀態位元 zr 和 ng。

Hack ALU 可以執行各種算術和邏輯運算，包括：

x + y

x - y

y - x

0

1

-1

x

y

!x

!y

-x

-y

x+1

y+1

x-1

y-1

x&y

x|y

zr 輸出為 1 表示 out 為 0。

ng 輸出為 1 表示 out 為負數。

時間的表示：

在數位系統中，時間是離散的，由一系列的時脈週期組成。

時脈： 一個產生一系列規律脈衝信號的元件，用於同步數位電路的運作。在硬體模擬器中，時脈可以用圖標或指令來表示。

暫存器：

一種可以儲存一定數量位元的電路。

暫存器的值在每個時脈週期更新一次。

n 位元暫存器： 一個可以儲存 n 個位元的暫存器。

n 位元暫存器的行為可以用真值表來描述。

n 位元暫存器可以用 n 個 1 位元暫存器來實現。

隨機存取記憶體 (RAM)：

一種允許以任何順序存取任何記憶體位置的記憶體。

RAM 可以使用邏輯閘和 DFF 來實現。

RAM 具有地址輸入、數據輸入、數據輸出和控制輸入（例如載入）。

RAM 的行為可以用真值表來描述。

計數器：

一種可以計數的電路，在每個時脈週期遞增其值。

計數器具有載入、遞增和重置輸入。

計數器的行為可以用狀態圖來描述。

Hack 指令集

Hack 計算機的指令集非常精簡，只有兩種指令類型：

A 指令 (地址指令)： 用於將一個常數值載入到 A 暫存器中。

語法：@xxx，其中 xxx 是一個非負整數。

語義：A ← xxx

副作用：會選擇 RAM[A] (表示為 M) 和 ROM[A]。

C 指令 (計算指令)： 用於執行計算，並將結果儲存到指定的暫存器中。

語法：dest = comp ; jump

dest：目標暫存器，可以是 A、D、M，或其組合。

comp：計算表达式，可以使用各種算術和邏輯運算符，例如 +、-、&、| 等。

jump：跳轉條件，可以是 JGT、JEQ、JGE、JLT、JNE、JLE 或 JMP。

語義：根據 comp 和 jump 的值，執行計算並更新指定的暫存器，以及可能跳轉到程式中的另一個位置。

符號程式設計

控制： 使用跳轉指令來控制程式的執行流程。

變數： 使用符號名稱來表示記憶體位置，讓程式更容易閱讀和理解。

標籤： 使用符號名稱來標記程式中的特定位置，以便跳轉指令可以使用。

Hack 語言

符號語言： 使用助記符號來表示指令和操作數，更容易被人類理解。

二進制語言： Hack 計算機實際執行的語言，由 0 和 1 組成。

輸出： 使用 Screen 記憶體映射將資料顯示在螢幕上。

輸入： 使用 Keyboard 記憶體映射從鍵盤讀取輸入。

CPU 的運作基於一個稱為提取-執行循環的基本循環：

1.

提取 (Fetch)： 從指令記憶體中提取下一條要執行的指令。

2.

解碼 (Decode)： 將指令解碼成 CPU 可以理解的操作碼和運算元。

3.

執行 (Execute)： 執行指令指定的操作。

Hack CPU 是 Hack 計算機的核心元件，負責執行指令和控制計算機的運作。Hack CPU 的主要組成部分包括：

ALU (算術邏輯單元)： 執行算術和邏輯運算。

暫存器： 儲存資料和中間結果，包括：

A 暫存器： 儲存記憶體地址或常數值。

D 暫存器： 儲存資料值。

程式計數器 (PC)： 儲存下一條要執行的指令的地址。

控制單元： 根據指令的解碼結果產生控制信號，控制 CPU 各個組成部分的運作。

輸入/輸出 (I/O)

Hack 計算機的輸入/輸出裝置透過記憶體映射來實現，也就是將 I/O 暫存器映射到特定的記憶體地址空間。

鍵盤： 使用 Keyboard 記憶體映射來讀取鍵盤輸入。

螢幕： 使用 Screen 記憶體映射將資料顯示在螢幕上。

記憶體

Hack 計算機的記憶體系統包含以下組成部分：

指令記憶體 (Instruction Memory)： 儲存程式指令。

資料記憶體 (Data Memory)： 儲存程式資料和變數。

RAM：用於儲存程式執行期間的資料。

Screen 記憶體映射：用於顯示螢幕內容。

Keyboard 記憶體映射：用於讀取鍵盤輸入。