Bitboard

From Wikipedia, the free encyclopedia

This article **does not cite any references or sources**. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. *(March 2008)*

A **bitboard** is a data structure commonly used in computer systems that play board games.

A bitboard, often used for boardgames such as chess, checkers, othello and word games, is a specialization of the bit array data structure, where each bit represents a game position or state, designed for optimization of speed and/or memory or disk use in mass calculations. Bits in the same bitboard relate to each other in the rules of the game often forming a game position when taken together. Other bitboards are commonly used as masks to transform or answer queries about positions. The "game" may be any game-like system where information is tightly packed in a structured form with "rules" affecting how the individual units or pieces relate.

Short description [edit]

Bitboards are used in many of the world's highest-rated chess playing programs such as Houdini, Stockfish, and Critter. They help the programs analyze chess positions with few CPU instructions and hold a massive number of positions in memory efficiently.

Bitboards allow the computer to answer some questions about game state with one logical operation. For example, if a chess program wants to know if the white player has any pawns in the center of the board (center four squares) it can just compare a bitboard for the player's pawns with one for the center of the board using a logical AND operation. If there are no center pawns then the result will be zero.

Query results can also be represented using bitboards. For example, the query "What are the squares between X and Y?" can be represented as a bitboard. These query results are generally pre-calculated, so that a

Wikipedia
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Deutsch
Français
日本語
Српски / srpski
Tagalog
Edit links

program can simply retrieve a query result with one memory load.

However, as a result of the massive compression and encoding, bitboard programs are not easy for software developers to either write or debug.

History [edit]

The bitboard method for holding a board game appears to have been invented in the mid-1950s, by Arthur Samuel and was used in his checkers program. The method was published in 1959 as "Some Studies in Machine Learning Using the Game of Checkers" in the IBM Journal of Research and Development.

For the more complicated game of chess, it appears the method was independently rediscovered later by the Kaissa team in the Soviet Union in the late 1960s, published in 1970 "Programming a computer to play chess" in Russ. Math. Surv. , and again by the authors of the U.S. Northwestern University program "Chess" in the early 1970s, and documented in 1977 in "Chess Skill in Man and Machine".

Description for all games or applications [edit]

Main article: Bit field

A bitboard or bit field is a format that stuffs a whole group of related boolean variables into the same machine word, typically representing positions on a board game. Each bit is a position, and when the bit is positive, a property of that position is true. In chess, for example, there would be a bitboard for black knights. There would be 64-bits where each bit represents a chess square. Another bitboard might be a constant representing the center four squares of the board. By comparing the two numbers with a bitwise logical AND instruction, we get a third bitboard which represents the black knights on the center four squares, if any. This format is generally more CPU and memory friendly than others.

General technical advantages and disadvantages [edit]

Processor use [edit]

Pros [edit]

The advantage of the bitboard representation is that it takes advantage of the essential logical bitwise operations available on nearly all CPUs that complete in one cycle and are fully pipelined and cached etc. Nearly all CPUs have AND, OR, NOR, and XOR. Many CPUs have additional bit instructions, such as finding the "first" bit, that make bitboard operations even more efficient. If they do not have instructions well known algorithms can perform some "magic" transformations that do these quickly.

Furthermore, modern CPUs have instruction pipelines that queue instructions for execution. A processor with multiple execution units can perform more than one instruction per cycle if more than one instruction is available in the pipeline. Branching (the use of conditionals like if) makes it harder for the processor to fill its pipeline(s) because the CPU can't tell what it needs to do in advance. Too much branching makes the pipeline less effective and potentially reduces the number of instructions the processor can execute per cycle. Many bitboard operations require fewer conditionals and therefore increase pipelining and make effective use of multiple execution units on many CPUs.

CPUs have a bit width which they are designed toward and can carry out bitwise operations in one cycle in this width. So, on a 64-bit or more CPU, 64-bit operations can occur in one instruction. There may be support for higher or lower width instructions. Many 32-bit CPUs may have some 64-bit instructions and those may take more than one cycle or otherwise be handicapped compared to their 32-bit instructions.

If the bitboard is larger than the width of the instruction set, then a performance hit will be the result. So a program using 64-bit bitboards would run faster on a real 64-bit processor than on a 32-bit processor.

Cons [edit]

Some queries are going to take longer than they would with perhaps arrays, so bitboards are generally used in conjunction with array boards in chess programs.

Memory use [edit]

Pros [edit]

Bitboards are extremely compact. Since only a very small amount of memory is required to represent a position or a mask, more positions can find their way into registers, full speed cache, Level 2 cache, etc. In this way, compactness translates into better performance (on most machines). Also on some machines this might mean

that more positions can be stored in main memory before going to disk.

Cons [edit]

For some games writing a suitable bitboard engine requires a fair amount of source code that will be longer than the straight forward implementation. For limited devices (like cell phones) with a limited number of registers or processor instruction cache, this can cause a problem. For full-sized computers it may cause cache misses between level one and level two cache. This is a potential problem—not a major drawback. Most machines will have enough instruction cache so that this isn't an issue.

Chess bitboards [edit]

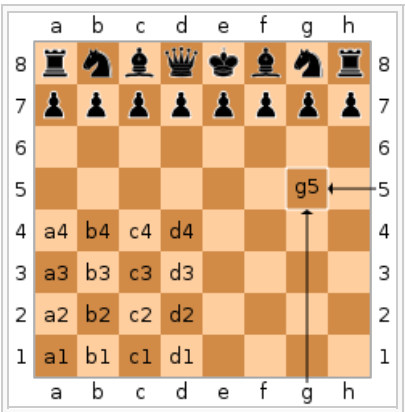
Standard [edit]

The first bit usually represents the square a1 (the lower left square), and the 64th bit represents the square h8 (the diagonally opposite square).

There are twelve types of pieces, and each type gets its own bitboard. Black pawns get a board, white pawns, etc. Together these twelve boards can represent a position. Some trivial information also needs to be tracked elsewhere; the programmer may use boolean variables for whether each side is in check, can castle, etc.

Constants are likely available, such as WHITE_SQUARES, BLACK_SQUARES, FILE_A, RANK_4 etc. More interesting ones might include CENTER, CORNERS, CASTLE_SQUARES, etc.

Examples of variables would be WHITE_ATTACKING, ATTACKED_BY_PAWN, WHITE_PASSED_PAWN, etc.



Algebraic notation

Rotated [edit]

"Rotated" bitboards are usually used in programs that use bitboards. Rotated bitboards make certain operations more efficient. While engines are simply referred to as "rotated bitboard engines," this is a misnomer as rotated boards are used in *addition* to normal boards making these hybrid standard/rotated bitboard engines.

These bitboards rotate the bitboard positions by 90 degrees, 45 degrees, and/or 315 degrees. A typical bitboard will have one byte per rank of the chess board. With this bitboard it's easy to determine rook attacks across a rank, using a table indexed by the occupied square and the occupied positions in the rank (because rook attacks stop at the first occupied square). By rotating the bitboard 90 degrees, rook attacks across a file can be examined the same way. Adding bitboards rotated 45 degrees and 315 degrees produces bitboards in which the diagonals are easy to examine. The queen can be examined by combining rook and bishop attacks. Rotated bitboards appear to have been developed separately and (essentially) simultaneously by the developers of the DarkThought and Crafty programs.[*citation needed*]

Magics [edit]

Magic move bitboard generation is a new and fast alternative to rotated move bitboard generators.[*citation needed*] These are also more versatile than rotated move bitboard generators because the generator can be used independently from any position.[*citation needed*] The basic idea is that you can use a multiply, right-shift hashing function to index a move database, which can be as small as 1.5K. A speedup is gained because no rotated bitboards need to be updated, and because the lookups are more cache-friendly.[*citation needed*]

Other bitboards [edit]

Many other games besides chess benefit from bitboards.

- In Connect Four, they allow for very efficient testing for four consecutive discs, by just two shift+and operations per direction.
- In the Conway's Game of Life, they are a possible alternative to arrays.
- Othello/Reversi (see the Reversi article).
- In word games, they allow for very efficient generation of valid moves by simple logical operations.

See also [edit]

- Bit array
- Bit field
- Bit manipulation
- Bitwise operation
- Board representation (chess)
- Boolean algebra (logic)
- Instruction set
- Instruction pipeline
- Opcode
- Bytecode

External links [edit]

Calculators [edit]

- 64 bits representation and manipulation

Checkers [edit]

- Checkers Bitboard Tutorial by Jonathan Kreuzer

Chess [edit]

Articles [edit]

- Programming area of the Beowulf project
- Heinz, Ernst A. How DarkThought plays chess. *ICCA Journal*, Vol. 20(3), pp. 166-176, Sept. 1997
- Laramee, Francois-Dominic. Chess Programming Part 2: Data Structures.
- Verhelst, Paul. Chess Board Representations
- Hyatt, Robert. Chess program board representations
- Hyatt, Robert. Rotated bitmaps, a new twist on an old idea
- Frayn, Colin. How to implement bitboards in a chess engine (chess programming theory)
- Pepicelli, Glen. *Bitfields, Bitboards, and Beyond* -(Example of bitboards in the Java Language and a discussion of why this optimization works with the Java Virtual Machine (www.OnJava.com publisher: O'Reilly 2005))
- Magic Move-Bitboard Generation in Computer Chess. Pradyumna Kannan

Code examples [edit]

- [1] The author of the Frenzee engine had posted some source examples.
- [2] A 155 line java Connect-4 program demonstrating the use of bitboards.

Implementations [edit]

Open source [edit]

- Beowulf Unix, Linux, Windows. Rotated bitboards.
- Crafty See the Crafty article. Written in straight C. Rotated bitboards in the old versions, now uses magic bitboards.
- GNU Chess See the GNU Chess Article.
- Stockfish UCI chess engine ranking second in Elo as of 2010
- Gray Matter C++, rotated bitboards.
- KnightCap GPL. ELO of 2300.
- Pepito C. Bitboard, by Carlos del Cacho. Windows and Linux binaries as well as source available.
- Simontacci Rotated bitboards.

Closed source [edit]

- DarkThought Home Page

Othello [edit]

- A complete discussion of Othello (Reversi) engines with some source code including an Othello bitboard in C and assembly.
- Edax (computing) See the Edax article. An Othello (Reversi) engine with source code based on bitboard.

Word Games [edit]

- Overview of bit board usage in word games.

Categories: Game artificial intelligence | Video board games | Computer chess | Bit data structures