mattrisASM: An Assembly Adventure

CS 2253 – Final Project
Matthew Kenneth Peterson (ID 3719754)

Idea

mattris.c
(silly *Tetris* clone from CS 2263)

Nintendo Entertainment System (NES)

Approach

- **Compile C directly for the NES, and pray it runs**

  X Bad Idea: cc6502 compiler is poorly optimized!

- **Suck it up and write it in 6502 assembly**

  X Bad Idea: I don’t know assembly yet, let alone for an ancient computer architecture!

- **Simplify first in C, convert to x86 assembly by hand under the guise of a productive school project, *then* finally tweak for 6502 assembly**

  Best idea I can come up with!

C Simplification Process

- Use global variables and pointers
- Trim down excess “ncurses” calls
- Implementing “goto” loops

```
for (int k = j; k > 1; k-- )
{
        board[k] = board[k-1];
}
```

“for” loop

```
k = i;

_clearLineLoop:

        board[k] = board[k-1];
        k--;

if (k > 1) goto _clearLineLoop;
```

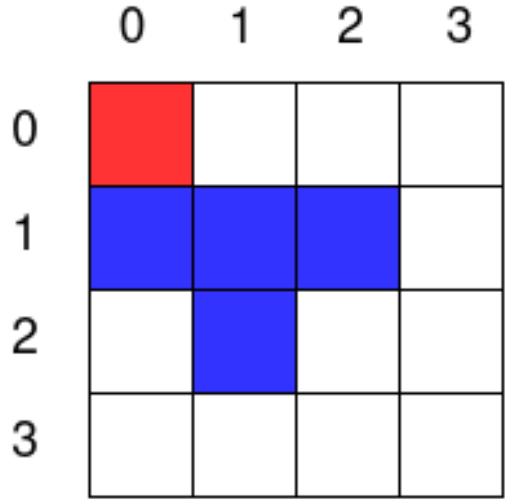
“goto” loop

Another Problem: The NES is weak!

	Average Personal Computer	NES
Instruction Set	Intel x86-64	MOS Technologies 6502
Working Memory	**8589934592 bytes**	**2048 bytes**
Stack Space	**As much as needed**	**Maximum 256 bytes**
Processing Speed	**3.00 GHz**	**0.001GHz**
Processor Cores	**6 to 8**	**1**
General Purpose Registers	**16 x 64-bit**	**2 Logic, 1 Math, all 8-bit**

Optimizing: Focus on Memory

- **Old “mattris” represented block as a set of coordinate points**
- **Example: T block**



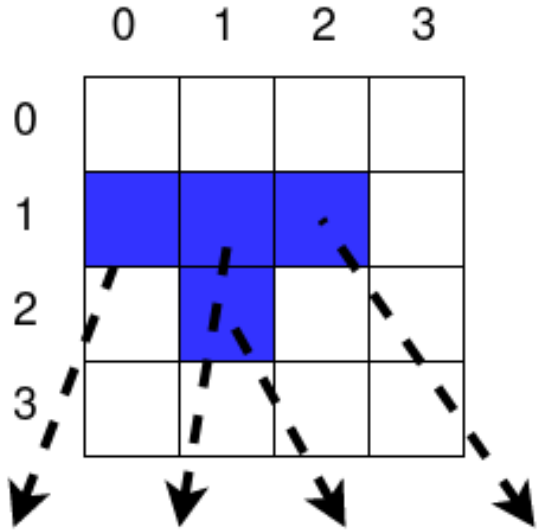
Integer (32 bit) Anchor X
Integer (32 bit) Anchor Y
Struct x4:
    Integer (32 bit) Block Chunk relative X
    Integer (32 bit) Block Chunk relative Y
    Pointer (64 bit) Next Block Chunk
+

576 bits, or 72 bytes, per block

The “TetrInteger”

- **Because each relative block-chunk coordinate lies between 0 and 3, we can represent them with the binary numbers (00, 01, 10, 11).**
- **By “packing” the converted result together, we get a single integer value which accounts for all of the important block data. This is effectively a custom data type.**
- **Original pointers are now unnecessary as all of the data is kept together.**

Example: T Block



Decimal : (0, 1) (1, 1) (1, 2) (2, 1)

Binary : (00,01) (01,01) (01,10) (10,01)

Bit Pack : 0001 0101 0110 1001

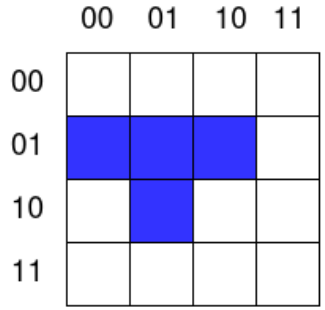
Once the bits have been packed, we may represent the entire block without the anchor in a single 16-bit unsigned integer!

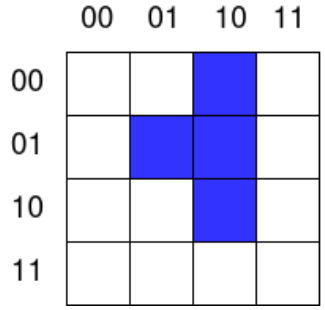
Using the TetrInteger

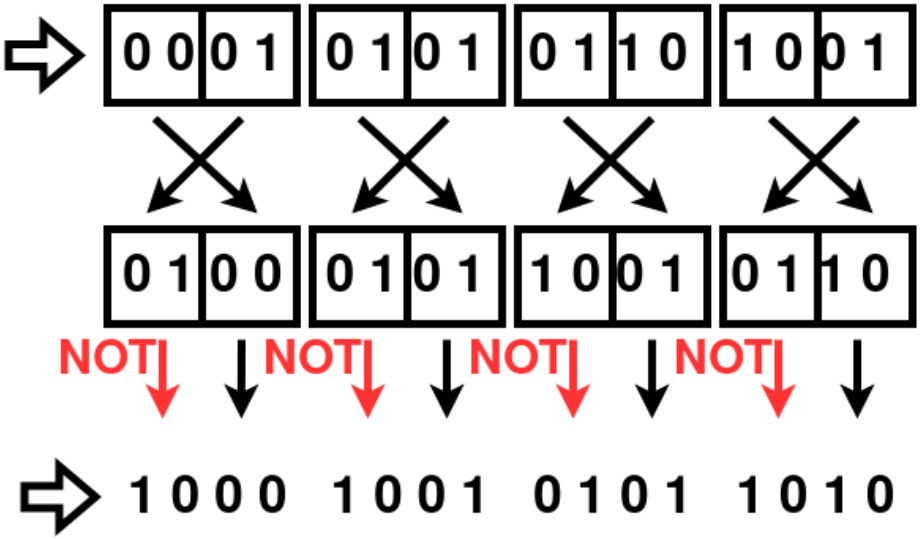
- The original implementation took 72 bytes per block. The new one only needs 3 unsigned 16-bit integers (2 for anchor point and 1 for block data), which is just 6 bytes! Much better for NES' 256 bytes of working/stack memory.
- Using bit-packing means that to use any data, one must use logical bitwise operations. This is perfect for an assembly oriented program, though is admittedly complicated and hard to follow
- Allows for some interesting algorithmic approaches

ASM Algorithm Example: Block Rotation

- **90-degree coordinate rotation follows (x,y) -> (-y,x)**







Assembly Re-Writes

- **Assembly code implementation in my program happened in tandem with the TetrInteger, due to the new techniques it allowed**
- **Works great with inline as well as external assembly, both of which I have implemented**
- **Not yet gotten to NES assembly, but that’s a journey for another day**

Assembly Rewrite: Example

To access the coordinates of a block chunk, I use a series of masks and shifts.

C Code

```
// 0x3 is a 2-bit mask.
resultX = temp & 0x3;
temp >>= 2;
resultY = temp & 0x3;
temp >>= 2;
```

Inline Assembly

```
asm volatile
(
        ".intel_syntax noprefix;"

        "mov eax, ecx ;"
        "and eax, 0x3 ;"
        "shr ecx, 0x2 ;"

        "mov ebx, ecx ;"
        "and ebx, 0x3 ;"
        "shr ecx, 0x2 ;"

        ".att_syntax ;"

        : "=a" ( resultX ), "=b" ( res
        : "a" (0), "b" (0), "c" ( temp
        : "cc"
);
```

Thank you for listening to me ramble :)

References:

- Wikipedia for NES image
- Microsoft Paint for diagrams
- Dr. Kim for teaching me assembly
- GCC documentation for inline-assembly related knowledge
- Over-abundances of caffeine for the TetrInteger idea