

COMP310: Legacy Game Systems

2: De-make culture

Research journal check-in



NES hardware



Nintendo Entertainment System (NES)



Nintendo Entertainment System (NES)

- ▶ Released in **1983**



Nintendo Entertainment System (NES)



- ▶ Released in **1983**
- ▶ Sold as the **Famicom** (Family Computer) in Japan

Nintendo Entertainment System (NES)



- ▶ Released in **1983**
- ▶ Sold as the **Famicom** (Family Computer) in Japan
- ▶ Nearly **62 million** units sold worldwide

Nintendo Entertainment System (NES)



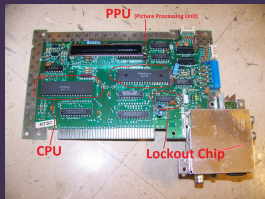
- ▶ Released in **1983**
- ▶ Sold as the **Famicom** (Family Computer) in Japan
- ▶ Nearly **62 million** units sold worldwide
- ▶ Biggest selling game: **Super Mario Bros**

Nintendo Entertainment System (NES)



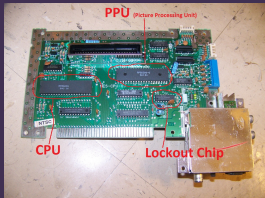
- ▶ Released in **1983**
- ▶ Sold as the **Famicom** (Family Computer) in Japan
- ▶ Nearly **62 million** units sold worldwide
- ▶ Biggest selling game: **Super Mario Bros**
- ▶ Credited with reviving the games industry after the **video game crash** of the early 80s

Nintendo Entertainment System (NES)



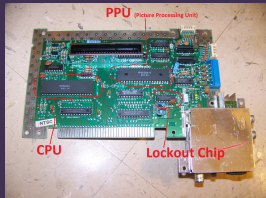
Nintendo Entertainment System (NES)

- CPU: Ricoh 2A03 (closely based on MOS 6502)

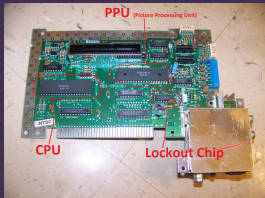


Nintendo Entertainment System (NES)

- ▶ CPU: Ricoh 2A03 (closely based on MOS 6502)
- ▶ Picture Processing Unit (PPU): Ricoh RP2C02 or RP2C07

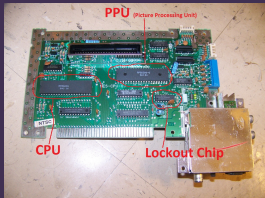


Nintendo Entertainment System (NES)



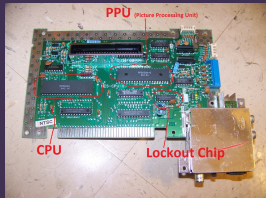
- ▶ CPU: Ricoh 2A03 (closely based on MOS 6502)
- ▶ Picture Processing Unit (PPU): Ricoh RP2C02 or RP2C07
- ▶ RAM: 2 kilobytes for CPU + 2 kilobytes for PPU

Nintendo Entertainment System (NES)



- ▶ CPU: Ricoh 2A03 (closely based on MOS 6502)
- ▶ Picture Processing Unit (PPU): Ricoh RP2C02 or RP2C07
- ▶ RAM: 2 kilobytes for CPU + 2 kilobytes for PPU
- ▶ Cartridge ROM: up to 1 megabyte, but typically less

Nintendo Entertainment System (NES)



- ▶ CPU: Ricoh 2A03 (closely based on MOS 6502)
- ▶ Picture Processing Unit (PPU): Ricoh RP2C02 or RP2C07
- ▶ RAM: 2 kilobytes for CPU + 2 kilobytes for PPU
- ▶ Cartridge ROM: up to 1 megabyte, but typically less
- ▶ Screen resolution: 256×240

Technical limitations

Technical limitations

- ▶ Around **2270 CPU cycles per frame**

Technical limitations

- ▶ Around **2270 CPU cycles per frame**
- ▶ Only **2 kilobytes** of writable memory to work with

Technical limitations

- ▶ Around **2270 CPU cycles per frame**
- ▶ Only **2 kilobytes** of writable memory to work with
- ▶ 6502 instruction set is limited

Technical limitations

- ▶ Around **2270 CPU cycles per frame**
- ▶ Only **2 kilobytes** of writable memory to work with
- ▶ 6502 instruction set is limited
 - ▶ 8-bit integer maths only

Technical limitations

- ▶ Around **2270 CPU cycles per frame**
- ▶ Only **2 kilobytes** of writable memory to work with
- ▶ 6502 instruction set is limited
 - ▶ 8-bit integer maths only
- ▶ The following are possible but need implementing as subroutines:

Technical limitations

- ▶ Around **2270 CPU cycles per frame**
- ▶ Only **2 kilobytes** of writable memory to work with
- ▶ 6502 instruction set is limited
 - ▶ 8-bit integer maths only
- ▶ The following are possible but need implementing as subroutines:
 - ▶ > 8 bit numbers

Technical limitations

- ▶ Around **2270 CPU cycles per frame**
- ▶ Only **2 kilobytes** of writable memory to work with
- ▶ 6502 instruction set is limited
 - ▶ 8-bit integer maths only
- ▶ The following are possible but need implementing as subroutines:
 - ▶ > 8 bit numbers
 - ▶ Multiplication

Technical limitations

- ▶ Around **2270 CPU cycles per frame**
- ▶ Only **2 kilobytes** of writable memory to work with
- ▶ 6502 instruction set is limited
 - ▶ 8-bit integer maths only
- ▶ The following are possible but need implementing as subroutines:
 - ▶ > 8 bit numbers
 - ▶ Multiplication
 - ▶ Division

Technical limitations

- ▶ Around **2270 CPU cycles per frame**
- ▶ Only **2 kilobytes** of writable memory to work with
- ▶ 6502 instruction set is limited
 - ▶ 8-bit integer maths only
- ▶ The following are possible but need implementing as subroutines:
 - ▶ > 8 bit numbers
 - ▶ Multiplication
 - ▶ Division
 - ▶ Fractional numbers

Graphical limitations

Graphical limitations

- ▶ Display is made up of **sprites** and **background**

Graphical limitations

- ▶ Display is made up of **sprites** and **background**
- ▶ Sprites:

Graphical limitations

- ▶ Display is made up of **sprites** and **background**
- ▶ Sprites:
 - ▶ Maximum 64 on screen

Graphical limitations

- ▶ Display is made up of **sprites** and **background**
- ▶ Sprites:
 - ▶ Maximum 64 on screen
 - ▶ Maximum 8 on the same scanline (horizontal line)

Graphical limitations

- ▶ Display is made up of **sprites** and **background**
- ▶ Sprites:
 - ▶ Maximum 64 on screen
 - ▶ Maximum 8 on the same scanline (horizontal line)
 - ▶ 8×8 pixels, 3 colours + transparency

Graphical limitations

- ▶ Display is made up of **sprites** and **background**
- ▶ Sprites:
 - ▶ Maximum 64 on screen
 - ▶ Maximum 8 on the same scanline (horizontal line)
 - ▶ 8×8 pixels, 3 colours + transparency
 - ▶ Can flip horizontally or vertically

Graphical limitations

- ▶ Display is made up of **sprites** and **background**
- ▶ Sprites:
 - ▶ Maximum 64 on screen
 - ▶ Maximum 8 on the same scanline (horizontal line)
 - ▶ 8×8 pixels, 3 colours + transparency
 - ▶ Can flip horizontally or vertically
 - ▶ No rotation, scaling etc.

Graphical limitations

- ▶ Display is made up of **sprites** and **background**
- ▶ Sprites:
 - ▶ Maximum 64 on screen
 - ▶ Maximum 8 on the same scanline (horizontal line)
 - ▶ 8×8 pixels, 3 colours + transparency
 - ▶ Can flip horizontally or vertically
 - ▶ No rotation, scaling etc.
- ▶ Background:

Graphical limitations

- ▶ Display is made up of **sprites** and **background**
- ▶ Sprites:
 - ▶ Maximum 64 on screen
 - ▶ Maximum 8 on the same scanline (horizontal line)
 - ▶ 8×8 pixels, 3 colours + transparency
 - ▶ Can flip horizontally or vertically
 - ▶ No rotation, scaling etc.
- ▶ Background:
 - ▶ Made of 8×8 pixel tiles

Graphical limitations

- ▶ Display is made up of **sprites** and **background**
- ▶ Sprites:
 - ▶ Maximum 64 on screen
 - ▶ Maximum 8 on the same scanline (horizontal line)
 - ▶ 8×8 pixels, 3 colours + transparency
 - ▶ Can flip horizontally or vertically
 - ▶ No rotation, scaling etc.
- ▶ Background:
 - ▶ Made of 8×8 pixel tiles
 - ▶ 32×32 blocks must share the same 4-colour palette

Graphical limitations

- ▶ Display is made up of **sprites** and **background**
- ▶ Sprites:
 - ▶ Maximum 64 on screen
 - ▶ Maximum 8 on the same scanline (horizontal line)
 - ▶ 8×8 pixels, 3 colours + transparency
 - ▶ Can flip horizontally or vertically
 - ▶ No rotation, scaling etc.
- ▶ Background:
 - ▶ Made of 8×8 pixel tiles
 - ▶ 32×32 blocks must share the same 4-colour palette
 - ▶ Limitations on types of scrolling

`https:
//wiki.nesdev.com/w/index.php/Limitations`

Examples of NES games

<https://youtu.be/um-GMygsRg4>

De-makes



De-makes

“purposedly built as an interpretation of how the game may have been, were it conceived and produced during a previous hardware or software generation”

<https://tvtropes.org/pmwiki/pmwiki.php/Main/VideogameDemake>

Constrained development task

Constrained development task

- ▶ Develop a NES de-make of a “modern” game

Constrained development task

- ▶ Develop a NES de-make of a “modern” game
- ▶ **Next week:** give a 5-minute pitch for your de-make

Constrained development task

- ▶ Develop a NES de-make of a “modern” game
- ▶ **Next week:** give a 5-minute pitch for your de-make
- ▶ Make sure you consider **scope** and **technical limitations** carefully

Constrained development task

- ▶ Develop a NES de-make of a “modern” game
- ▶ **Next week:** give a 5-minute pitch for your de-make
- ▶ Make sure you consider **scope** and **technical limitations** carefully
- ▶ Focus on a single key mechanic

Constrained development task

- ▶ Develop a NES de-make of a “modern” game
- ▶ **Next week:** give a 5-minute pitch for your de-make
- ▶ Make sure you consider **scope** and **technical limitations** carefully
- ▶ Focus on a single key mechanic
- ▶ Focus on gameplay, not graphics or content

Developing for the NES



Tools

Tools

► An **emulator**

Tools

- ▶ An **emulator**
 - ▶ Recommended: FCEUX
 - ▶ <http://www.fceux.com/>

Tools

- ▶ An **emulator**
 - ▶ Recommended: FCEUX
 - ▶ <http://www.fceux.com/>
- ▶ An **assembler**

Tools

- ▶ An **emulator**

- ▶ Recommended: FCEUX
- ▶ <http://www.fceux.com/>

- ▶ An **assembler**

- ▶ Recommended: NESASM
- ▶ <https://github.com/edpowley/nasasm/releases>

Tools

- ▶ An **emulator**
 - ▶ Recommended: FCEUX
 - ▶ <http://www.fceux.com/>
- ▶ An **assembler**
 - ▶ Recommended: NESASM
 - ▶ <https://github.com/edpowley/nasasm/releases>
- ▶ A **sprite editor**

Tools

- ▶ An **emulator**

- ▶ Recommended: FCEUX
- ▶ <http://www.fceux.com/>

- ▶ An **assembler**

- ▶ Recommended: NESASM
- ▶ <https://github.com/edpowley/nesasm/releases>

- ▶ A **sprite editor**

- ▶ Recommended: YY-CHR
- ▶ <https://www.romhacking.net/utilities/119/>

Tools

- ▶ An **emulator**

- ▶ Recommended: FCEUX
- ▶ <http://www.fceux.com/>

- ▶ An **assembler**

- ▶ Recommended: NESASM
- ▶ <https://github.com/edpowley/nesasm/releases>

- ▶ A **sprite editor**

- ▶ Recommended: YY-CHR
- ▶ <https://www.romhacking.net/utilities/119/>

- ▶ A **text editor**

Live coding videos

Live coding videos

► bit.ly/comp310

Live coding videos

- ▶ `bit.ly/comp310`
- ▶ I expect you to work through these **in your own time**

Live coding videos

- ▶ bit.ly/comp310
- ▶ I expect you to work through these **in your own time**
- ▶ Timetabled workshops are mostly for **working on your projects** and getting **support** (although there will also be a bit of taught material)

Let's jump in!

Let's jump in!

- ▶ `http://nintendoage.com/forum/messageview.cfm?catid=22&threadid=7974`

Let's jump in!

- ▶ `http://nintendoage.com/forum/messageview.cfm?catid=22&threadid=7974`
- ▶ `Download controller.zip`

Exercise

Modify `controller.asm` so that all of Mario moves left and right, not just the back of his head...