ECE 498 – Linux Assembly Language Lecture 8

Vince Weaver

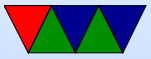
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

11 December 2012

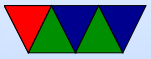
Video Game Programming

- My personal gateway into assembly programming (and programming in general). Apple BASIC books in library back in the day.
- One of the areas where often low-level/assembly programming still used, though less so now that gaming hardware is so powerful.
- Having a multi-year fixed target for programming also makes assembly more attractive.



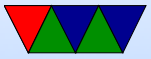
Video Game Needs

- Some way to get the image you want on screen
- Some way to read user input



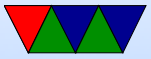
Optional features

- Some way to double-buffer (draw offscreen)
- Some way to generate sound/music



2d-gameplay

- Early days
- Easy to program
- Despite brief resurgence with phones/tablets/etc, moving back to 3d



TB1-game

- DOS VGA Mode 13h

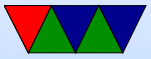
  ```
  mov #0x13, ax ; int 0x10
  ```

- Linear framebuffer at A000:0000, 320x200 (256 color)

- Can choose colors from larger palette

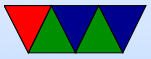
  ```
  mov 0x3c8,dx; mov COLOR,al; out dx,al, ...
  ```

- Fading with palette. Wait for retrace.

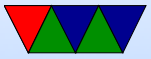


2d-gameplay

- Draw background, score, etc
- Draw character (ship)
- Draw enemies, weapons, explosions ("sprites")
- Copy to display
- Check keypress, act on keypress
- Move everyone, collision detect

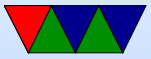


- Loop



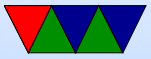
Porting to Linux

- In theory "framebuffer" allows linear framebuffer
- In practice stuck with X most of the time
  You do not want to code to X11 in assembly. Bad enough in C.
- Use SDL abstraction library. More or less works and is reasonably fast. Also works OSX/Windows



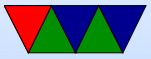
3d-gameplay

"Guinea Pig Adventure" demo (written in OpenGL)



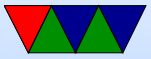
3d-gameplay

- More complicated. Lots of linear algebra
- Can use libraries
- GPU / Video card sometimes more powerful than CPU. Can be programmed at low level too, though usually not at the assembly level (hidden by manufactures)
  Why? To keep their trade secrets, also let them change the underlying implementation.



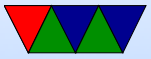
Video Game Systems – 7th generations

- PS/3 (2006) – 3.2GHz Cell(Power) / NVIDIA
- Xbox360 (2005)– 3.2GHz Power / ATI
- Wii (2006) – 729MHz Power / ATI
- DS (2004) – 67MHz armv4t
- PSP (2004) – 333MHz MIPS R4000



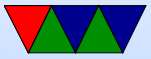
Video Game Systems – 6th generation

- PS/2 (2000) – 294MHz MIPS "Emotion Engine" / "Graphics Synthesizer"
- Xbox (2001) – 733MHz Pentium III / Custom
- Gamecube (2001) – 485MHz Power / ATI
- Dreamcast (1998) – 200MHz SuperH SH4 / PowerVR
- GameBoyAdvance (2001) – 16MHz armv4t



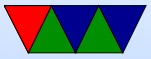
Video Game Systems – 5th generation

- Playstation (1994) – 33MHz MIPS R3000
- N64 (1996) – 93MHz MIPS NECVR4300
- Sega Saturn (1994) – 28Mhz SH-2



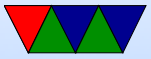
Video Game Systems – 4th generation

- TurboGrafx (1987) – 65SC02
- Sega Genesis (1988) – m68k/z80
- Super Nintendo (1990) – 3.58MHz 65c816
- Neo Geo (1991) – 12MHz m68k / z80
- Gameboy (1989) – z80-like custom
- Atari Lynx (1989) – 6502



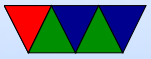
Video Game Systems – 3rd generation

- NES (1983) – 6502 compatible
- Sega Master System (1985) – z80 Clone



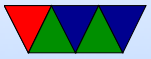
Video Game Systems – 2nd generation

- Atari 2600 (1977) – 6507
- Mattel Intellivision (1979) – CP1610
- Colecovision (1982) – z80



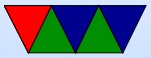
Video Game Systems – 1st generation

- Pong (1975) – custom
- Magnavox Odyssey (1975) – custom

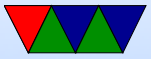


Super Nintendo

- “5A22” processor, mostly a 65c816, 3.58MHz, 128kB RAM
- NMI on vertical blanking interrupt
- screen position IRQ
- DMA for block transfers, also auto transfer during horizontal blanking
- multiply/divide hardware

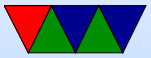


- matrix hardware (3d-esque and color effects)
- 2-32Megabit ROMs
- Sound : 8-bit Sony SPC700 controlling 8-channel Sony DSP, 512kbit buffer
- Video: 64kB VRAM. 15-bit color (with 256 palette), 32768 colors on screen at once. 256x224 graphics typical (higher possible but flicker). 128 sprites on screen at once.



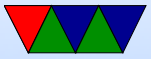
Super Nintendo – Video Refresh

- On CRT screens the display pauses in drawing as the electron beam is turned off to return to beginning of screen (horizontal) and longer to return to top (vertical)
- You try to do display update work then
- SNES provides DMA and IRQs during these times for this purpose.



Super Nintendo

- 7 different background modes
- Use Mode 0
  - 32x32 grid of “tiles”
  - Tile can be one of 1024 in VRAM
  - vhopppcc cccccccc – vert flip, horiz flip, prioirity, color (pal), tile
  - this gives 256x256, only 256x224 visible, scrolling
- Sprites

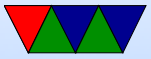


Super Nintendo – Sprites

- Can display 128 sprites (only 32 per scanline)
- 16-colors but can choose palette
- Info stored in OAM (Object Attribute Memory)
- First table

  xxxxxxxx yyyyyyyy cccccccc vhoopppc

  x, y, sprite, vert flip, horiz flip, priority, color, most sig bit

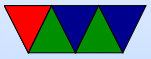


- Second 32-byte table

  xsxsxsxsxsx

  most sig x bit, toggle bit

-



Super Nintendo – Hello World Example

- Uses same cross-compiler as Apple II
- Creates a ROM image for use with SNES emulator (snes9x, bsnes, zsnes)
- Code complicated. No O/S, no built in text handling, so lots of code to just put a few tiles on the screen.