Programowanie pod ZX Spectrum Next

Osobiste doświadczenie RCLa z VVG

Speccy.pl Party 2025

Trochę kontekstu

- Jestem RCL/Virtual Vision Group (oraz /Suspend na PC)
- Programuję pod ZX Spectrum od roku 1994
  - W assemblerze od roku 1995
  - …z przerwami …dużymi (1994-1999, 2005, 2012, 2024-...?)
- Wyrosłem na klonach i “wannabe-pecetach”
  - Didaktik Skalica, Profi+
  - TR-DOS
  - 1MB RAM
- Porzuciłem aktywne programowanie Spectruma w roku 1999
  - Ale nie przestałem programować ogólnie

Więcej kontekstu

- ZX Spectrum Next jest rozwinięciem Spectruma
  - Widziałem wiele takich zaczynając od końca lat `90, lecz ta jest IMHO udana
- FPGA… ale dalej to jest sprzęt
- 2MB RAM
  - są nieliczne już wersje z 1MB
- 28 Mhz
  - jak się chce, ale kto by nie chciał?
- “Rozszerzone” Z80
  - MUL !!! oraz inne dogodne instrukcje
- DMA, CTC, TurboSound (3x AY), sprajty, tilesy, 256 kolorów…

O czym jest ten wykład?

- Chcę przybliżyć ludziom platformę
  - Ale zakładam jakąś tam znajomość programowania pod Z80 oraz Spectruma / klony
- Chcę się podzielić osobistym doświadczeniem
  - Nie należy traktować tego jako oficjalnego poradnika
  - Nie wszystko rozkminiłem, nie wszystko robię jak “należy”
- Po co?
  - Żeby nie trzeba było wystawiać prac pod Nexta jako Wild :)
    - Dzięki, Tygrysie!
  - Podoba mi się ta platforma i chcę by więcej scenowców ją wsparło
    - Choć w sumie po co mi konkurencja…

O czym NIE jest ten wykład?

- Po co ludzie wymyślają nowe rzeczy
- Dlaczego Next a nie XYZ (TS-Config, MB-03 etc)
- Czy FPGA to jest sprzęt
- Czy Next to jest retro
- Ile Ci zapłacili za reklamę

Początki - formaty wykonywalne

- Szypowalne
  - .NEX
    - Gry, dema
  - .DOT
    - Intra

- Jest więcej, lecz ich nie potrzebowalem
  - https://wiki.specnext.dev/File_Formats

- Next jest **platformą wybitnie przyjazną programiście asemblera**
- Ma dwa podstawowe formaty
  - .NEX - ideologicznie podobny do .SNA lub .Z80
  - .DOT - ideologicznie podobny do .COM z PC lub CP/M
    - Max 8KB kodu wczytywanego pod adres $2000 (8192), bez żadnych nagłówków
- Nie potrzebujesz żadnych loaderów i BASICa (choć BASIC ogólnie jest)
- Jak chcesz 256b intro - lepszego formatu niż DOT nie ma

Początki - format NEX

- Dla dem
- Łatwy w obsłudze
- NIE jest dla size-coding'a

- NEX jest bardziej rozwinięty niż SNA
  - Może zawierać np. loading screen oraz dodatkowe dane (do ręcznego wczytywania), zapisane na końcu pliku
  - W przeciwieństwie do SNA dystrybucja dema w NEX nie przyniesie Ci wstydu
  - Sjasm+ wspiera go natywnie od wersji sprzed paru lat

```
; adres startu i stosu
SAVENEX OPEN "helloworld.nex", START, $FF40
; wersja rdzenia (sprzętu)
SAVENEX CORE 3, 0, 0
; kolor borderu oraz flagi
SAVENEX CFG 7, 0, 1, 0
SAVENEX AUTO ; zgrać całą zapisaną pamięć
SAVENEX CLOSE
```

- Tu jest więcej: https://z00m128.github.io/sjasmplus/documentation.html#c_savenex

- **DUŻA WADA**: minimalny rozmiar jest około 16KB
  - chcesz mniej? Korzystaj z .DOT

Kompatybilność

- Domyślnie Next ma ekran oraz kolory Spectruma
- Spectrumowe mapowanie pamięci jest wspierane
- Dostęp do rejestrów Neksta jest możliwy przez porty

- Jak najprościej zacząć programować pod Next?
- Zamienić DEVICE ZXSPECTRUM128 na DEVICE ZXSPECTRUMNEXT oraz zapisać plik w formacie NEX
- O ile nie wywołujesz procedur z ROM, będzie działać tak samo
- Układ pamięci ekranowej jest taki sam
- Co więcej, można nawet “rozpoznać” Nexta z programu stricte spectrumowego (np. w .tap) i korzystać z dodatkowej funkcjonalności
  - Patrz mój kod na githubie: https://github.com/RCL/ZXSpectrumNextEtudes/
- Przenoszenie oprogramowania spectrumowego pod Next jest **bardzo łatwe**!

Oprogramowanie pod Next nie musi korzystać z dodatkowych możliwości i trybów graficznych.

Może on też być traktowany jako zastępca Pentagonu z 50Hz (albo nawet 60Hz).

Nowe możliwości - pamięć

- Zamiast $7FFD można korzystać z 8 rejestrów Next (portów) mapujących poszczególne 8 kilobajtowe obszary na banki
- Zmiana mapowania nie wymaga “psucia” BC (ani nawet A jeśli bank jest znany z góry).
- Tabela wektorów przerwań musi być wyrównana tylko dla 32 bajtów i nie musi zajmować 256 bajtów

- Praca z pamięcią jest łatwiejsza!

- Dostęp do pamięci jest przez okienka 8 KB
  - Można zostawić “kernel” ze stosem w adresach $E000-$FFFF i przełączać dolne 56KB
  - Lub na odwrót, przełączać tylko pamięć wideo z $E000, mając kod i dane w dolnych 56KB
  - Można przechowywać tabele z adresu $0000

- Przełączanie banków jest łatwe i szybkie
  - nextreg <rejestr_okienka>, <numer_banku>
  - nextreg <rejestr_okienka>, a

- Jest o *wiele* więcej pamięci (co najmniej 1MB, ale można po prostu się orientować na 2MB)

- Żadnej wolnej czy innej “specjalnej” pamięci - jest jak Pentagon

- Tabela wektorów przerwań nakłada mniej ograniczeń

Nowe możliwości - CPU

- Obliczanie adresów ekranu w CPU
- Dodawanie do rejestrów 16-bitowych
- Przesuwanie DE
- Mnożenie 8-bit
- Oraz wiele innych: https://wiki.specnext.dev/Extended_Z80_instruction_set

- Pisanie kodu pod Next jest również łatwiejsze (jak na 8-bitowca)!

- Instrukcje obliczania adresu w pamięci ekranowej ULA
  - PIXELAD - HL = screen_addr(E, D)
  - SETAE - A = (0x80 >> E)
  - PIXELDN - HL = next_scanline(HL)

- Instrukcje dodające do HL i DE - mocno ułatwiają chodzenie po tabelach / liniowym ekranie
  - ADD HL, 32
  - ADD DE, 2
  - ADD DE, A

- Przesuwanie DE na z góry nieznaną liczbę bitów
  - BSRA DE, B - DE = (DE >> B)

- Mnożenie (bez znaku) DE = D*E
  - MUL

- Testowanie (bezdestrukcyjny AND)
  - TEST $7

Ok super, ale jak to do cholery odpalić?

I to jest obecnie największą bolączką Nexta.

Emulatory

- Tylko dwa emulatory nadające się do użycia przez człowiekowatych. Pick your poison:
- CSpect
  - https://mdf200.itch.io/cspect
- ZEsarUX
  - https://github.com/chernandezba/zesarux/releases

- CSpect
  - Plusy
    - Najwierniej (z tego co wiem) odwzorowuje Next
    - Zintegrowany z debugerem i DeZog'iem
    - Odpala się na Macu i Linuksie (ponoć)
  - Minusy
    - **Szyfrowany** closed source i autor nie chce go podpisać, przez co jest oflagowywany przez antywirusy - **DUŻA WADA**
    - Autor zabrania rozpowszechniać CSpect bez jego zgody **- DUŻA WADA**. Chciałbym dodać zip z emulatorem i systemem "pod klucz" do tej prezentacji, lecz nie mogę
    - C#, więc wydajność jest nierówna
    - Krzywe AY
    - UX z epoki kamienia łupanego
- ZEsarUX
  - Kombajn, który nawet nie chce nazwać Nexta Nekstem
    - (szukajcie TBBlue)
  - UX jest o parę epok do przodu, gdzieś tak rok 1994
  - Nie korzystałem za bardzo, nie mogę się wypowiadać

Emulowanie

- Należy zrobić obraz systemu
- “Poprawne” emulowanie zakłada umieszczenie plików w tym obrazie i odpalenie stamtąd
- Odpalenie NEX’ów z komendy poleceń jest przez twórców widziane jako nadające się tylko do szybkich testów

- Setup jest trochę bólem.
- Zrobiłem to raz i kopiuję pomiędzy komputerami.
- Muszę w to jeszcze raz looknąć lecz zawsze jest coś ważniejszego niż “poprawny” setup, gdyż dla mojego use case *w większości* działa
- Nie zawracajcie sobie zbytnio tym głowy dopóki nie zaczniecie pisać gry lub innego oprogramowania co musi coś odczytywać lub zapisywać z dysku / na dysk.
- Po prostu odpalam za pomocą CSpectu podając mu plik .NEX w linijce poleceń
- **WADA** (nieduża ale jednak): odpalanie .dot w taki sposób nie jest możliwe. Zazwyczaj można to obejść zapisując DOT jako NEX podczas debugowania, ale różnice są i na to ostatnio się nadziałem.

Czas przestać straszyć i pokazać, jak w praktyce działa emulowanie oraz debugowanie (hint: nie jest źle).

Kwestie wydajnościowe

- Dalej jest to 8-bitowiec. 28 Mhz to jest 8x 3.5Mhz, ale instrukcje są dalej 8-bitowe i zajmują dużo cykli
- Przenosząc kod raytracing'owy z klasyki, spodziewałem się cudów. Natomiast nawet z 28Mhz i zamianą na MUL przyśpieszenie było około 9 krotne, z 40-paru do około 5 sekund.

- 28Mhz brzmi dużo ale jest to **tylko 8x szybsze** od klasyka. Większe rozmiary pamięci ekranów skutecznie "zjadają" uzyskaną przewagę
  - Przy ekranie 6912 lub 6144 bajtów jest całkiem fajny wzrost wydajności
  - Przy ekranie 12 KB, na dodatek podzielonym na części, jest gorzej
  - Przy ekranie 48 KB (256x192x8bpp) i wyżej wracamy do fillrate prawie spectrumowego

- Next ma dodatkowy **pomocniczy sprzęt**, lecz jest on przede wszystkim przydatny do klasycznych efektów 2D:
  - Sprajty
  - Tryb tilesowy
  - Podkład / nakład (overlay)

- Do dowolnych efektów pikselowych (3D, plasmy / rotozoomery) najbardziej nadaje się **tryb 128x96x4bpp** (LoRes / Radastan), gdzie ekran zajmuje 6144 bajtów.

- Można też **pozostać przy ekranie ULA**-i !

60 klatek na sekundę

- Next wspiera 60Hz. Jest to poniekąd bardziej przyszłościowy format, nawet w Europie
- O liczbie klatek można się dowiedzieć programistycznie, więc warto to IMHO wesprzeć
- Pominięcie każdego 6. update'u muzyki oraz numeru klatki / logiki jest akceptowalnym rozwiązaniem

- Pamiętajmy że Next może być **odpalony w 60Hz**.
  - IMHO należy nawet się nastawiać, że będzie, jak nie teraz to w 2030
  - 50Hz ekrany odchodzą, szczególnie wśród TV
  - Chciałoby się by nasze płynne animacje dalej były płynne w muzeach demosceny

- Najprostszym sposobem na wsparcie 60Hz jest pominięcie każdego 6. cyklu pętli głównej lub przerwania.
  - Muzyka może zacząć wyć, lecz jest to niezbyt zauważalne, szczególnie jak się o tym nie wie

- Innym sposobem jest programowanie timera na 50Hz i użycie tego do update-u muzyki oraz licznika klatek tudzież logiki, lecz pozostawienie synchronizacji z vblank do rysowania
  - Nie jest to aż tak skomplikowane jak to brzmi, zrobiłem to, ale jest to więcej zachodu

- Pamiętajmy, że w 60Hz jest **mniejsza o 17% liczba taktów** per klatka (z **560k** spada do **467k**)

Funkcjonalność zaawansowana

- DMA
- Przerwania CTC
- Reszta

- DMA jest dobrze opisane w dokumentacji. Ma jeden kanał, działa jak **szybszy LDIR**. Nie korzystałem jeszcze z DMA “w boju”, tylko testowałem.

- Ma tryb działania w tle, lecz **nie ma przerywania po ukończeniu** działania. Przeznaczony przeważnie do odgrywania sampli, spowalnia pamięć (ta sama zasada jak chip RAM na Amidze)

- CTC to jest **timer**, generuje przerwania o podanej częstotliwości (jako dzielnik częstotliwości systemowej). Jest w miarę proste w obsłudze, lecz trzeba pamiętać że HALT już nie będzie tożsamy z początkiem klatki -> to będzie trzeba dodatkowo sprawdzać.

- Przerwań liniowych, sprajtów, tilesów, Coppera jeszcze nie używałem. Można o nich poczytać w dokumentacji jak się chce, wydają się być dobrze opisane.

- Emulatory miewają problemy z taką funkcjonalnością, ale CSpect ma już ponoć większość z nich za sobą

Pomocne linki

- https://wiki.specnext.dev/
  - Wiki na różne tematy techniczne
- https://github.com/tomaz/zx-next-dev-guide
  - Przewodnik
- http://ped.7gods.org/Z80N_table_ClrHome.html
  - Tabela wszystkich instrukcji Z80N
- https://github.com/RCL/ZXSpectrumOpenSource/ oraz https://github.com/RCL/ZXSpectrumNextEtudes
  - Źródła moich dem, oraz oddzielne przykłady
- Jest więcej źródeł od **taylorza** oraz innych - poszukajcie na githubie
- https://discord.gg/UXVHCxuAWg - Discord Nexta po angielsku

Next ma większe moce ekspresyjne niż klasyczne Spectrum.

Ale dalej jest to maszyna 8-bitowa, pozwalająca na całkowite przejęcie kontroli nad nią.

Mam nadzieję, że będziecie mieli frajdę z programowania pod nią!

Pytania / uwagi

Discord: RCL1024

Mastodon:

rcl@mastodon.gamedev.place