Amiga 500 i wstęp do sekretów DMA

Marcin Nowosad (KaiN)
kain@piwnica.ws
github.com/tehKaiN

Przebieg wykładu

- Architektura komputera Amiga 500
- Problematyka cykli dostępu do pamięci
- Krótko o działaniu Denise i Coppera
- Narzędzia do analizy cykli DMA

Procesor i pamięć: prymitywny komputer

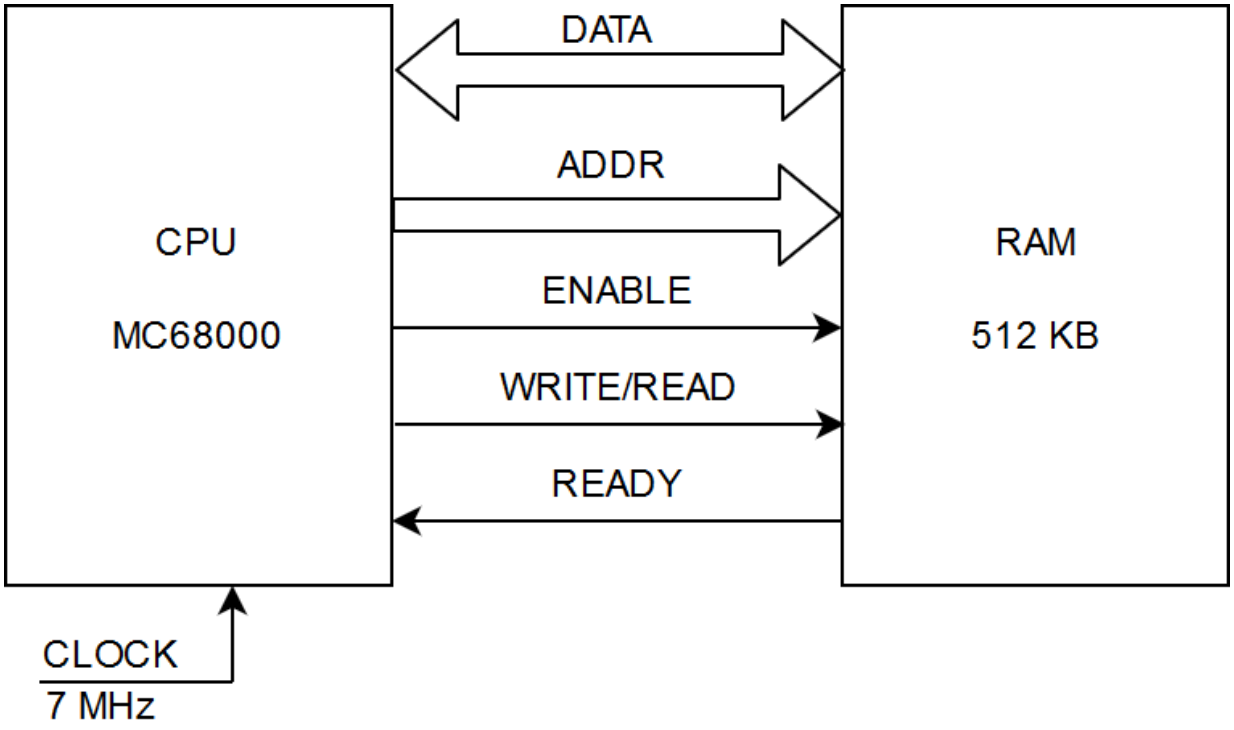
- Najważniejsze pojęcia związane z procesorem:
  - Rejestry
  - Instrukcje
  - Dostęp do pamięci

Prosty program na Amigę

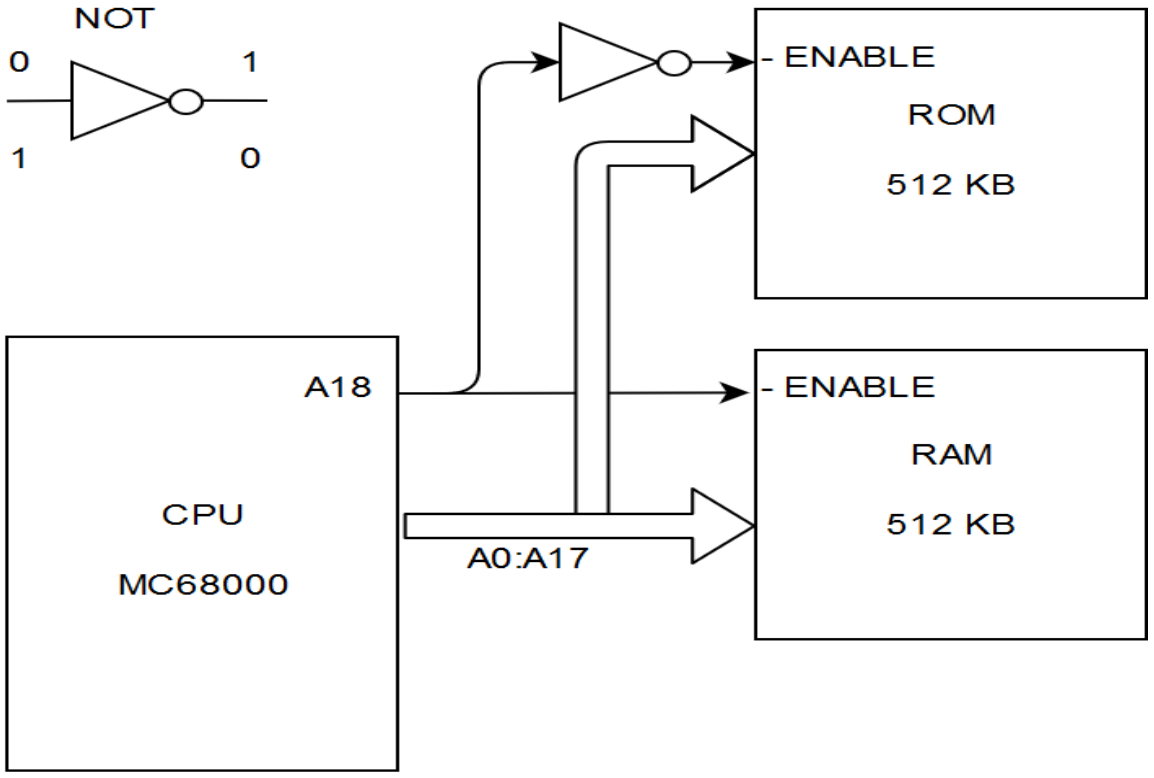
- Załaduj do rejestru „d0” liczbę 5
- Załaduj do rejestru „d1” liczbę 7
- Dodaj rejestry „d0” i „d1”
- Zawartość rejestru „d0” zapisz w pamięci pod adresem 7

☞ Potrzebujemy czegoś, co pozwoli CPU
na odczyt instrukcji i zapis danych

Komunikacja z RAMem

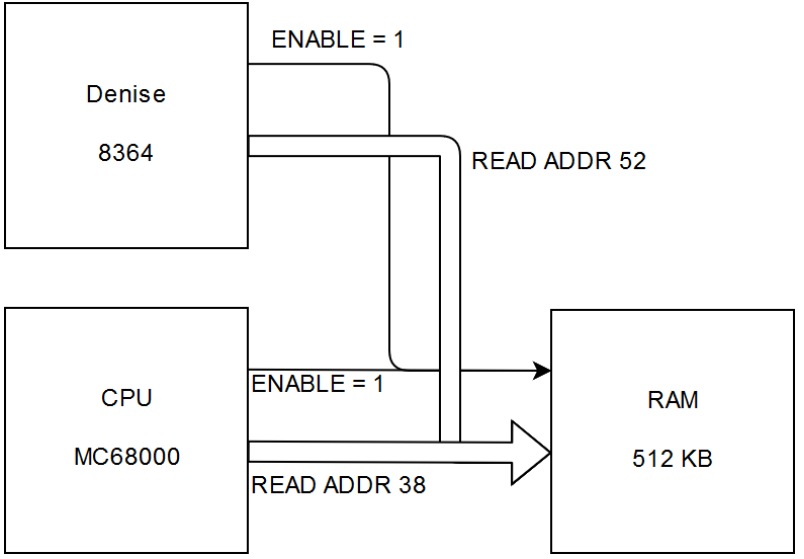


Ale przecież RAM jest na początku pusty...



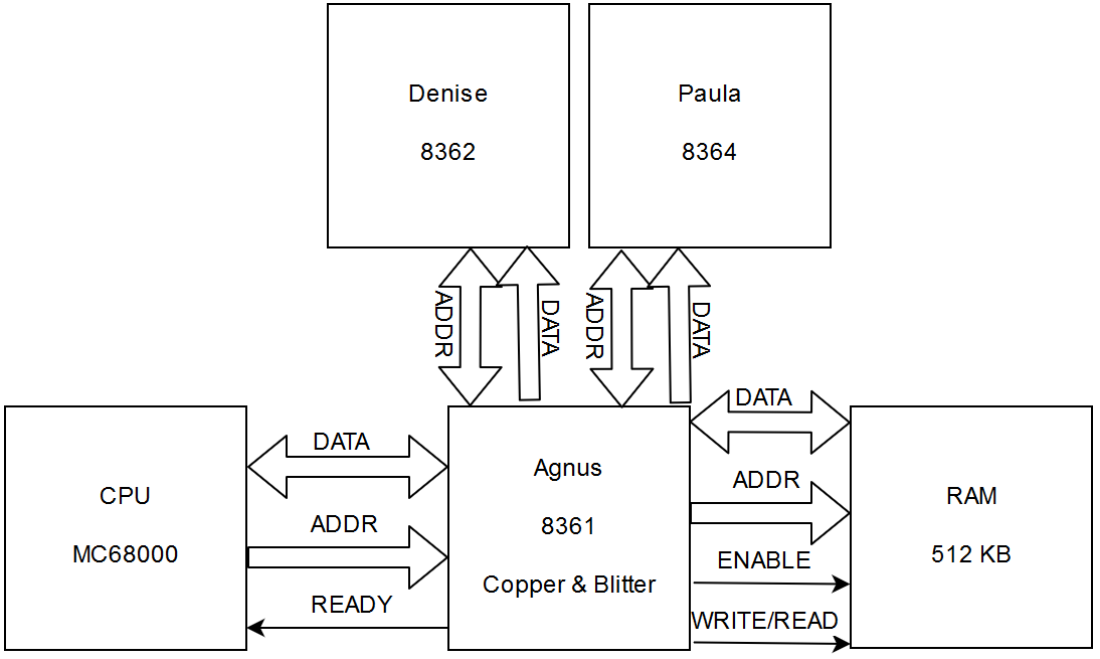
Wyświetlmy coś na ekranie

- Dołóżmy do układanki Denise
- Denise również potrzebuje dostępu do RAMu



Agnus jako kontroler pamięci

- Arbitracja dostępu do magistrali pamięci



Problem na horyzoncie

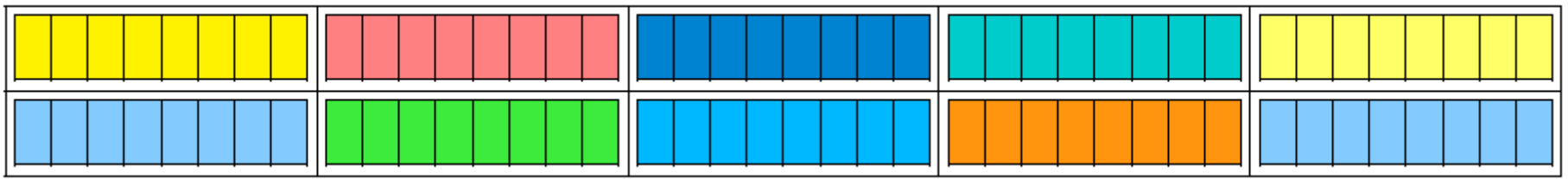
- Podzespoły potrzebujące dostępu do RAMu:
  - Paula – sample dźwiękowe
  - Denise – wyświetlanie grafiki, dane sprite’ów
  - Procesor – instrukcje, dane
  - Blitter – rysowanie linii, wypełnianie obszarów, boby
  - Copper – instrukcje
- Jak to wszystko ze sobą pogodzić?

Prędkość dostępu do pamięci w Amidze

- 1 dostęp do pamięci (slot) trwa tyle, co narysowanie 2 pikseli na ekranie
- Liczba slotów podczas jednej sekundy: 3 556 800
- Liczba slotów podczas jednej klatki obrazu (50Hz): 71 136
- Podział slotów każdemu po równo: 14 227 slotów
- Dużo czy mało? Czy tak można?

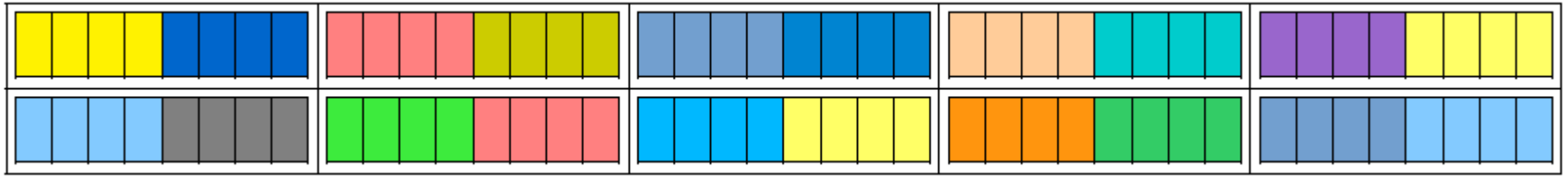
Wyświetlanie obrazu na ekranie

- Podejście najprostsze – tryb Chunky
- 8 bitów na piksel, odczyt co bajt, paleta 2^8 = 256 kolorów
- Zużycie RAMu: 320x256 = 81920 bajtów = 80KB
- Co zrobić gdy chcemy tylko np. 16 kolorów?



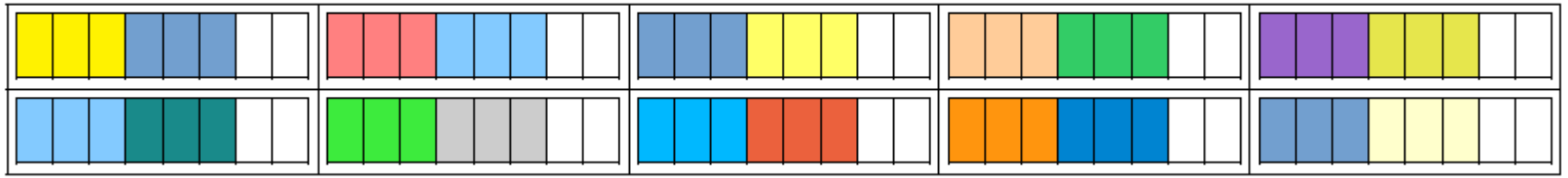
Wyświetlanie obrazu na ekranie

- 4 bitów na piksel, dwa kolory w bajcie, paleta $2^4 = 16$ kolorów
- Zużycie RAMu: $320 \times 256 \cdot 0.5 = 40960$ bajtów = 40KB
- Co zrobić gdy chcemy tylko np. 8 kolorów?



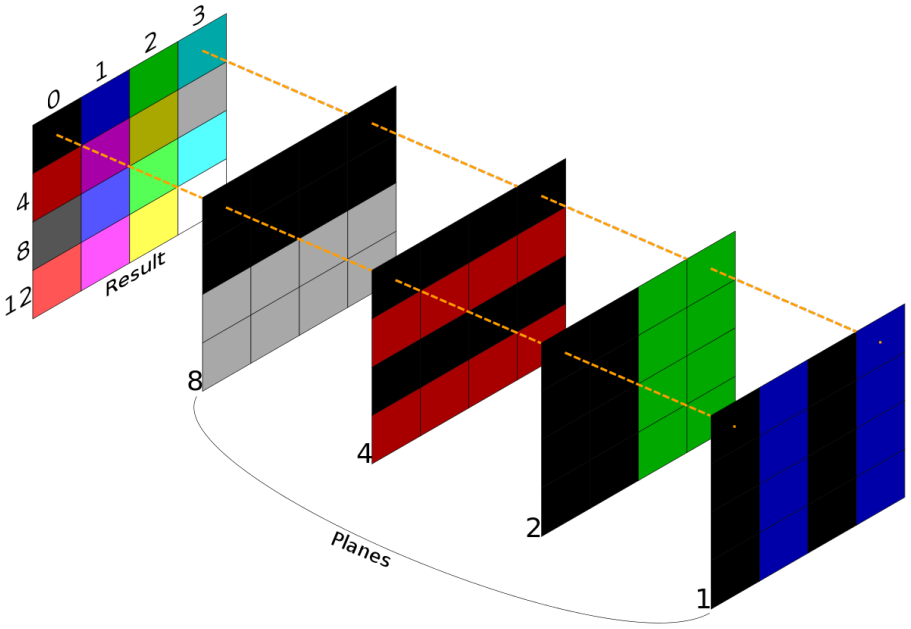
Wyświetlanie obrazu na ekranie

- 3 bity na piksel, dwa kolory w bajcie, paleta $2^3 = 8$ kolorów
- Duże zużycie RAMu: 320x256 · 0.5 = 40960 bajtów = 40KB
- Tryb chunky nie pozwala na wydajne przechowywanie obrazów w palecie 8, 32, 64 (i 128) kolorów



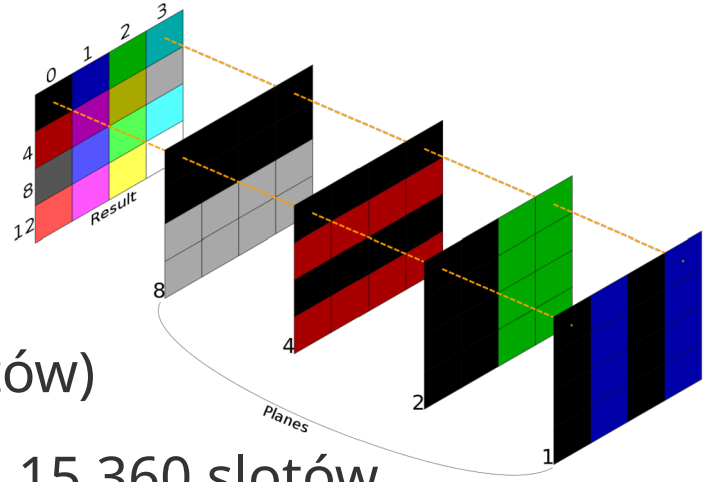
Tryb planar

- Przy 4 bitach na piksel: zamiast jednego obrazu 4 oddzielne 1bpp
- Każdy bitplane ma 320 bitów (40 bajtów) szerokości
- Jeden bitplane 320x256 zajmuje 40x256 = 10240 bajtów = 10 KB
- 3bpp: 3 · 10KB = 30KB
- 5bpp: 4 · 10KB = 40KB



Pobieranie bitplane’ów

- Denise czyta 2 bajty (16 bitów) z jednego bitplane’u naraz – 16px
- W przypadku 3 bitplane’ów Denise musi pobrać 3 takie porcje danych
- Jeden slot trwa tyle co wyświetlenie 2px
- W ciągu wyświetlenia 16px na ekranie jest 8 slotów – Denise zabierze 3 (38% slotów)
- Do wyświetlenia 320x256@3bpp potrzeba 15 360 slotów



Wizualizowanie zajętości cykli

- Amiga Hardware Reference Manual

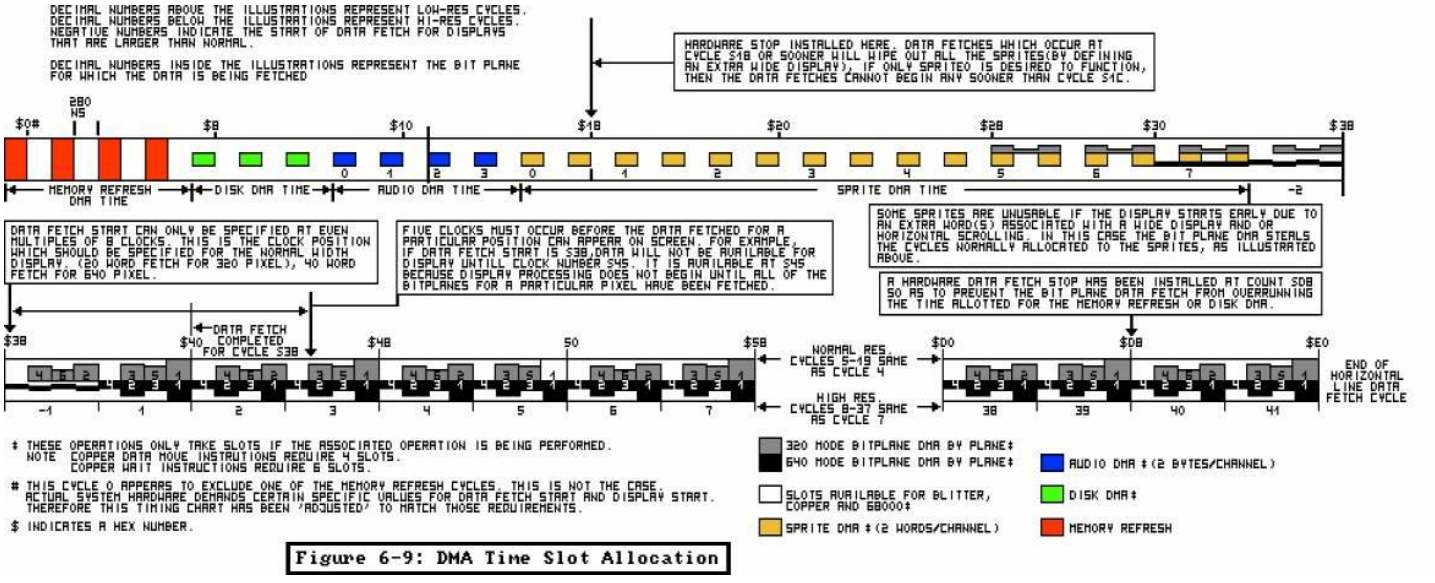


Figure 6-9: DMA Time Slot Allocation

Inne źródła wiedzy?

- Wzajemnie przeczące sobie stwierdzenia w HRM
- Masa uproszczeń nie dających pełnego obrazu sytuacji
- Posty na forach można policzyć na palcach dwóch dłoni

Narzędzia pomagające analizę zajętości cykli

- Stan na 2017:

Narzędzia pomagające analizę zajętości cykli

- Stan na 2018:
  - Ami-cycles
    Kod źródłowy: github.com/tehKaiN/ami-cycles
    Aktualna wersja: tehKaiN.github.io/ami-cycles

Demo ami-cycles

Prezentacja na żywo aplikacji

Materiały z prezentacji

Slajdy z prezentacji dostępne są pod adresem:

piwnica.ws/tworczosc/materialy/

Dziękuję za uwagę