Designskiss

TSEA83

Projekt Team Hjärncell 583

Grupp 34

Medlemmar:
Emir Hadzisalihovic(emiha868)
Christian Habib(chrha376)
Ilian Ayoub(iliay038)

7 mars 2017

1 Analys

- Vi ska klocka allt med 100 MHz ty nexyskortets systemklocka har samma frekvens. Eftersom vi ska använda oss av VGA-skärmen med upplösningen 640x480 pixlar där pixelklockan är på 25MHz. Systemet har då 4 klockcykler på sig att rita ut en pixel.

- I kravspecifikationen angav vi att spelplanen skulle vara 40x40 tiles stor där en tile är 4x4 pixlar stor. Vardera pixel skulle representeras i 4 färger (2-bit).Däremot skulle vi vilja ändra på kraven då vi insåg att det fanns utrymme för fler tiles eftersom vi hade räknat fel när vi skrev kravspecifikationen. Istället vill vi ha ett 80x60 tiles spelplan med 4x4 pixlar stora tiles där vardera pixel kan anta 2^3 färger (3 bit).

- Nexykortes består av 576 kbit(72000 bytes) blockram. Ett blockram är ett tvåportsminne där man kan skriva till en port samtidigt som man läser från ena porten. En blockram kan betraktas som en vanlig array och det är vad vi kommer använda oss av till lagring.[4]

- Eftersom vi har endast 8 färger samt ett få antal olika tiles(5 st) så borde vi ha tillräckligt mycket minne för att använda oss av dubbelbuffring till att räkna ut hur spelplanen ska se ut i nästa tick. Det intressanta är ormens rörelse och kollisionhantering mellan olika tiles. Ormen skulle man kunna representera som en länkad lista där varje tile av ormen pekar på nästa tile som är grannen. Listan är då länkad hela vägen fram till huvudet av ormen och huvudet ska då ha en riktning. Men här fastnar vi ett mjukvarutänk. Troligtvis finns det en bättre lösning.

- Monohögtalaren (bit banging) ska se till att göra ett ljud när ormen äter upp mat. Ett ljud uppstår om en bit växlar mellan låg och hög i en hög frekvens. Detta ska hårdvaruenheten högtalaren sköta, programmet ska bara säga till enheten när den ska tjuta.

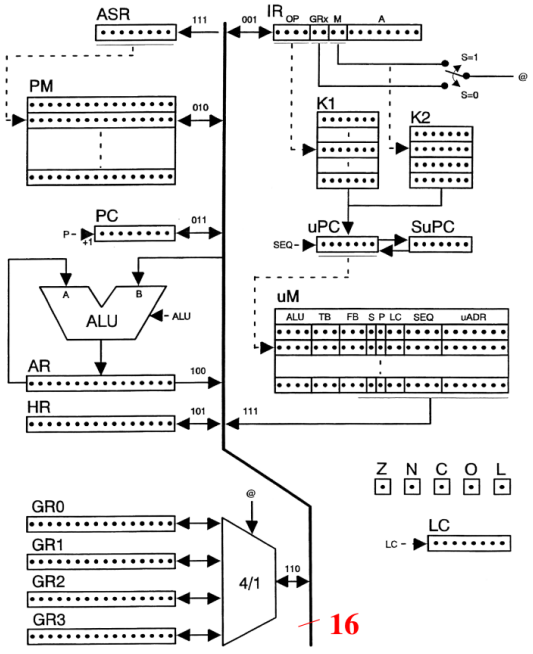
2 Vår konstruktion

2.1 CPU och instruktionsset

Vi hade tänkt använda oss av en mikroprogrammerad processor med kombinerad minne (data och instruktioner i samma minne).Vi ska ha tillräckligt många instruktioner för att kunna göra loopar och kolla förhållanden, såsom aritmetiska instruktioner, load, store, samt villkorliga och ovillkorliga hopp. Eftersom vi kommer ha 4800 tiles (alternativt 1600 om vi utgår från kravspec) så krävs det en addressrymd på $2^{13} = 8192$. Vi kommer att använda oss av följande adresseringsmoder: aboslut-, indirekt absolut,absolut indexerad och relativ addressering. Adresseringsmoden kan representeras med två bitar i instruktionen. Operandkoden hade vi tänkt oss vara 6 bitar lång. Då har vi plats med $2^6 = 64$ olika instruktioner. Därmed anser vi att en ordbredd på 32 bitar är lämpligt då OP-koden,addressrymden och adresseringsmoden själva tar upp 21 bitar och med en 32 bitar ordbredd får vi ytterligare plats med fler argument. CPU:ns främsta jobb är att läsa från bildminnet och rita om det samt är dess uppgift att flytta och läsa värden från I/O enheterna med hjälp av en bus. Processorn ska startas genom att ladda ett färdigt program, se avsnitt 2.4.

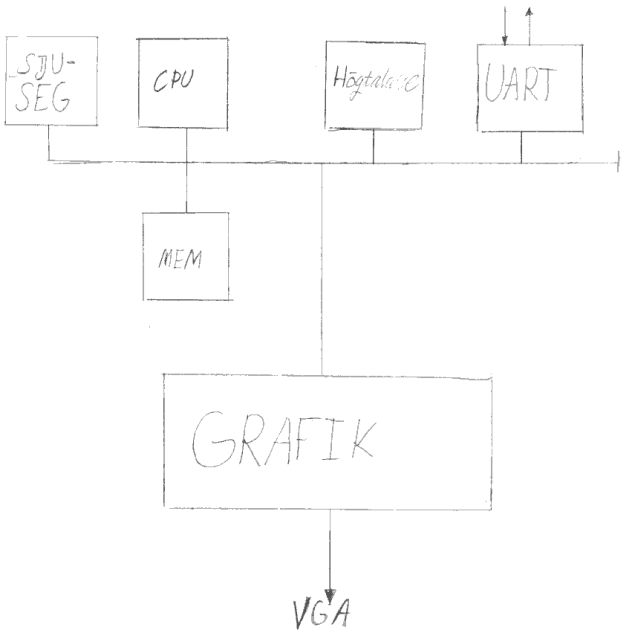
2.2 Schema av CPU

Vi har bestämt oss för att använda oss av samma design som Björn Lindskogs dator där vi kompletterear mikroaddressens bredd ut efter våra behov. Det är en dator som är simpel att begripa sig på, den är däremot inte pipelinad. En load instruktion tar upp till 6 klockcykler att utföra. Primärminnet motsvarar blocket 'mem' som återfinns i den grova blockschemat(se avsnitt 2.3). Om vi låter varje tick vara i 2 sekunder så borde Björn Lindkogs dator vara tillräckligt snabb för att utföra grovjobbet i att ändra om spelplanen.



Figur 1: *Björn Lindskogsdatorn direkt tagen ur föreläsning 3 [3]*

2.3 Grov Blockschema



2.4 Minnesanvändning

Nexyskortet har 576 kbit blockram vilket motsvarar 72000 bytes.[1][2] Spelets spelplan består av 80x60 tiles vilket motsvarar 4800 stycken tiles. Vi kommer att begränsa oss till 5 stycken olika tiles: hinder, mat, orm, ormhuvud, bakgrund. Med 8 färger får vi ett tileminne som är $5 * 16 * 3 = 30$ bytes stort. Vi kan ha ett separat tileminne m.en för tillfället lagrar vi det i program-minnet. Spelplanen kommer att ta upp $80 * 60 * 32 = 19200$ bytes, detta motsvarar storleken av bildminnet. Däremot om vi har ett seperat tileminne behöver vi endast ange 1 byte per cell i bildminnet istället för 8. Vi har då cirka 50000 bytes kvar att röra oss med.

2.5 Programmering

Själva spelet ska vara skriven i C och med hjälp av en assemblator(assembler) kan vi översätta programmet till maskinkod som är anpassad till vår CPU. Hur det översatta programmet ska föras över till programminnet är vi osäkra på. För tillfället säger vi att processorn kommer startas med att man laddar in programmet via UART.

2.6 I/O-enhter

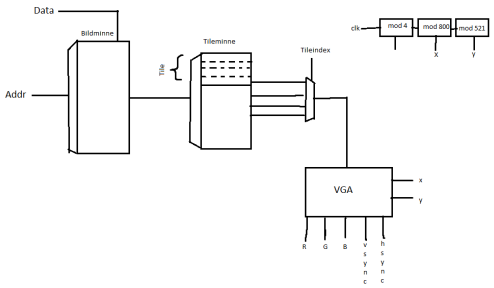
Alla enheter ska kunna kommunicera med varandra m.h.a parallela bussar där CPU-enheten är MASTER och resterande enheter är SLAVE.

2.7 Implementation av UART

Gällande Funktionaliteten av UART hade vi tänkt ha en styrenhet som lyssnar efter startbit av en knapp. Tangentbordet kommer skicka ett bitflöde serriellt.

När styrenheten får en startbit så ska den räkna antal pulser för att avgöra när en knapp har lästs in. Knappens värde skickas sedan ut till en LUT(lookup-table) som avgör om det är piltangenter eller knapparna WASD som har tryckts in, utgången av UARTen till bussen ändras efter varje knapptryck. Det ska även ske någon form av handskakning mellan UART:en och CPU:n för att klargöra när processorn har läst av en tangent.

2.8 Blockschema av Grafikenheten



Adressen pekar på vilken tile som man ska behandla. Adressen räknar på det sättet att den itererar fyra gånger över samma tile-rad, för att sedan inkrementera till nästa rad av tiles. I tile-minnet finns alla typer tiles sparade som pekas ut av bildminnet. Tile-indexet pekar på vilken rad i tilen som ska ritas ut av VGA:n. Tile-indexet räknas från 0 till 3, den räknar upp varje gång x har räknat till 800.

3 Komponentlista

1. UART kopplad tangentbord
2. Monohögtalare
3. Nexyskortet samt dess sjusegment
4. VGA-skärm

4 Milstolpe

Vår milstolpe är att efter halva terminen ska vi ha implementerat en dator med en fungerande assemblator. Vi ska i alla fall kunna visa upp att en spelplan med en orm ritas ut på skärmen.

Referenser

[1] Digilent. Nexys3 board reference manual. https://liuonline.sharepoint.com/sites/TSEA83/TSEA83-2017VT/CourseDocuments/Nexys3_rm.pdf, 2011. [Online; åtkomst 27 Februari 2017].

[2] Anders Nilsson. Föreläsning i grafik. https://liuonline.sharepoint.com/sites/TSEA83/TSEA83-2017VT/CourseDocuments/OH_grafik.pdf, 2017. [Online; åtkomst 27 Februari 2017].

[3] Anders Nilsson. Föreläsning om mikropogrammering ii. https://liuonline.sharepoint.com/sites/TSEA83/TSEA83-2017VT/CourseDocuments/OH_mikro2.pdf, 2017. [Online; åtkomst 6 Mars 2017].

[4] Anders Nilsson. Föreläsning om minnen och bussar. https://liuonline.sharepoint.com/sites/TSEA83/TSEA83-2017VT/CourseDocuments/OH_mem_buss.pdf, 2017. [Online; åtkomst 6 Mars 2017].