Designspecifikation Epik Snake

TSEA83, G48

version: 0.3

Edvard Wetind edvwe024@student.liu.se
Hugo Nilsson hugni385@student.liu.se
Christoffer Näs chrna581@student.liu.se
Mikael Lundgren miklu523@student.liu.se

Sammanfattning

Denna designspecifikation är menad att ge en grov översikt över implementationen av processorn i projektet. Utöver en mikro-programmerad processor kommer konstruktionen även vara kapabel till att avkoda signaler från en joystick. Processorn kommer även innehålla en VGA-motor.

Definitioner

Adresseringsmoder

Adresseringen i processorn kommer främst att ske på tre sätt: direkt, indirekt och relativ. Med dessa två moder möjliggörs smidig skrivning till och från programminnet.

Programminne

Två delar. En “ram del” som uppdateras under spelets gång, typ ormens längd.

En “instruktions del” som sparar instruktionerna och programmeringen för spelet.

Processor

Själva logiken i programmet. Lägger till frukter, förlänger ormen, tar hand om varje tick i spelet. Varje tick styrs av interrupt.

Grafikmotor

Skriver till skärmen vad som ska visas och var. Uppdateras varje tick.

Bildminne

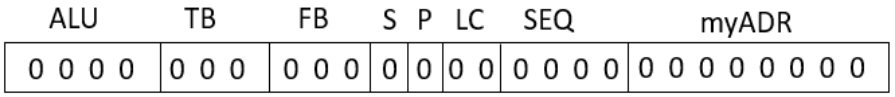
Sparar “bilder” på hur de olika “pixlarna” ska se ut. T.ex. vilken färg och hur ormen ska se ut.

Snake Controller

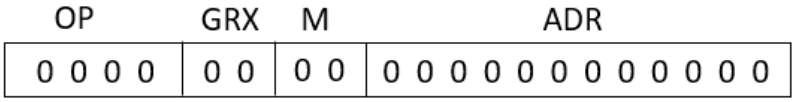
Tar hand om input och output från kontrollen. Avkodar standard-output:en till gallrad version med 4 bitar som representerar fyra riktningar. Dessa bitar använder sedan processorn för att flytta ormens huvudsegment.

CPU

Målet med projektet är att konstruera en mikroprogrammerad processor. Datorns mikroinstruktioner kommer att vara 26 bitar breda. Instruktionerna kommer att vara 20 bitar breda. Enbart direkt addressering kommer att användas. Se nästa sida för komplett instruktionsset.



Micro Instruktioner



Instruktionsformat

Instruktionsset

Init funktion

NOP	no operation
HALT	halt
LD	GRx := PM(A)
LDI	GRx := I
STORE	PM(A) := GRx
ADD	GRx := GRx + PM(A)
ADDI	GRx := GRx + I
AND	Rd <= Rd && Ra
ANDI	Rd <= Rd && constant
SUB	GRx := GRx - PM(A)
SUBI	GRx := GRx - I
LSR	GRx >> X times
BRA	jump to x
BNE	jump if Z=0
BSE	jump if N=1
BGE	jump if N=1 & O=1
BEQ	jump if Z=1
BPL	jump if N=0
BLT	jump if N=1 & O=1
CMP	GRx == PM(A)
CMPI	GRx == I
AND	GRx && PM(A)
ANDI	GRx && I
OR	GRx || PM(A)
ORI	GRx || I

Grafik

Spelet kommer att använda sig av enkel grafik för att rita ut objekten. Ormen och äpplet är svart, resten vitt. Ormens positioner samt äpplets är sparat i programminnet.

Grafiken kommer hanteras av en VGA-Motor, som ritar ut ormen och frukten på dess plats. Spelets upplösning kommer att vara 640x480, då varje tile är 8x8 pixel. Detta resulterar i en spelplan på 40x30 tiles. Äpplet är 1 tile stor. Ormen är från början 2 tiles lång, men ökar för varje frukt den äter.

I/O-enheter

Vår I/O-enhet är en joystick med SPI ansluts till Nexys-3 för att låta användaren styra ormen.

Minnesanvändning

Datorns minne består av programminnet. Det är här alla instruktioner ligger lagrade. Hämtning av instruktioner sker via ett instruktions-register, i blockschemat betecknat IR.

Våra olika minnen kommer att behöva ha storleken:

- Joystick: 4-bitars register.
- SR, statusregistret kommer att behöva rymma 5 flaggor men vi gör den till 8 bitar för enkelhetens skull.
- Instruktionsregistret och program minnet kommer att behöva vara 28 bitar brett.
- Programminnet kommer att behöva 4-bitar för I/0.
- BILDMINNE: 40*30*(000000-SPACE,UP,DOWN,LEFT,RIGHT,APPLE,HEAD) för grafik och status.

Programmering

För programmeringen av spelet så kommer en (game)loop behöva programmeras i PM. Där finns olika “sektioner” som körs beroende på vad som händer i spelet/vilka flaggor som är satta. Det vill säga, assemblerkod med subrutiner. Vi kommer att hårdkoda i mikroprogrammering i datorn men använda oss av assembler instruktioner för att ta fram

mikrokoden. Vi återanvänder VGA-motorn som vi använde i VGA-labben för att realisera VGA-motorn.

Timing

Processorn klockas i 100 Mhz en uppdateringsfrekvens på 2fps uppnås genom att ta processor klockan på 100 Mhz och applicera en mjukvarubaserad räknare som räknar upp till 50 M. Kör nästa “tick”s instruktioner på en gång efter en uppdatering till skärmen. Men väntar tills skärmen har uppdaterats innan den kör nästa “tick”s instruktioner. Vi kommer att ha en vänte loop i programmet, för att vänta på bilduppdatering. Alltså kommer programmet att behöva vänta in nästa tick.

Milstolpar

Som halvtids mål har vi en färdigbyggd dator som klarar av någonting grafiskt och I/O relaterat.

Blockscheman