DESIGNSPEC

åL

TSEA83

Elliot Norlander, ellno907
Jennifer Santos, jensa682
Jacob Sjölin, jacsj573
Elin Rydebrink, eliry213

Version 1, 28/2-2023

Sammanfattning

Denna designspecifikation är menad att ge en övergripande bild av konstruktionen av en mikroprogrammerad processor. Utöver processorn innehåller också konstruktionen en avkodningsmodul för inläsning av en joystick samt en VGA-motor för generering av VGA-baserad bilddata.

Implementation

Det kommer användas flera olika typer av hårdvara. Till exempel kommer FGPA-kortet vara ansvarig för VGA-motorn, åL-controller, programminnet och bildminnet. Joysticken kommer göra det möjligt att styra ålen. Det kommer även användas en VGA-monitor för att visualisera spelet.

CPU

Processorn ska kunna hantera 32-bitars instruktioner. Med en grafikupplösning på 600 x 600 pixlar, krävs det 10 bitar för att beskriva en viss ruta. Därutöver skall det också finnas plats för instruktionsnotering (AND, ADDI, etc.) samt flaggor och andra register. Med de ovan nämnda specifikationerna är 16-bitars instruktioner inte tillräckligt och därav har beslutet fattats att 32-bitars instruktioner kommer användas i denna processor. Angående adresseringsmoder skall det finnas stöd för direkt, indirekt samt registerbaserad adressering. Processor kommer att inneha ett förladdat program.

Grafik

Planen är att programmet kommer använda bildminnesdriven grafik. För att få in färg, används bitmapgrafik med två bitar. På så sätt kommer fyra olika färger kunna användas. Värdet för varje pixel kommer att sparas bitvis i minnet.

Spelplanen kommer vara uppdelad i 15 x 15 bildrutor. Där varje bildruta har 40 x 40 pixlar och varje pixel kan anta någon av de fyra färgerna. För att generera bilddatan kommer tile-baserad grafik användas. Tanken är använda tiles för att hantera den rutiga bakgrunden, ålen samt fisken som ålen ska äta upp.

Då det är tile-baserad grafik som används behöver VGA-motorn inte synkroniseras med CPUn. Tanken är att VGA-motorn löpande genererar bilddatan, och att CPUn styr när bilddatan faktiskt ska skickas från bildminnet till VGA-monitorn.

I/O-enheter

För att tolka input från joysticken kommer det behövas en adressavkodare som kan läsa av joystickens signaler. Den avkodade datan kommer sedan att lagras i ett avsatt register. Outputen till VGA-monitorn kommer att hanteras av VGA-motorn i kombination med CPUn. Inputen från joysticken kommer att lagras i CPUn och bilddatan kommer att sparas i grafikminnet.

Minnesanvändning

Konstruktionen kommer bestå av ett bildminne (32 bitar), instruktionsregister (26 bitar), programminne (32 bitar), flaggregister (1-bitar). Enligt beräkningar kommer bildminnet vara 15*15 = 225 bitar ≈ 29 bytes stort. IR kommer ha 26 instruktionsrader, programminnet

estimeras till maximalt 2048 rader och flaggregistret kommer vara 4 rader. CPUn kommer läsa och skriva i bildminnet och därför kommer Block-RAM att användas.

Programmering

Planen är att skapa ett set av instruktioner med hjälp av mikroprogrammering. Instruktionerna skall sedan användas i syfte att skapa de individuella delarna av åL.

åL-controller är spelets main-loop. Kollisionshantering kontrolleras här och annan funktionalitet som möjliggör ett fullt fungerande spel i enlighet med spelets regler.

Bildminnet sparas i block-RAM eftersom CPU-n måste kunna skriva och läsa i detta minne. Inuti bildminnet sparas ett tile-nummer som kopplas till varje enskild bildruta i spelplanen.

I VGA-motorn sparas tileminnet med information om hur varje tile ska se ut. VGA-motorn genererar bild- och synksignaler, vilket resulterar i att spelplanen ritas upp varje uppdateringsfrekvens.

Timing

För att spelet ska ha en rimlig hastighet är tanken att implementera en vänteloop. Genom att ladda in talet 25M i ett register, och sedan subtrahera ett från detta tal tills vi når noll, kan man med processors klocka på 100 MHz och en subtraktionsoperation som tar 1 cykel, fördröja spelflödet med 0,25 sekunder. Denna vänteloop kan sedan utföras i följd för att fördröja med till exempel 0,5 och 0,75 sekunder. Vi tänker att denna implementeras i en separat timermodul.

Instruktionsuppsättning

Ett set av mikroprogrammerade instruktioner kommer skapas. Se figur 1.

Milstolpe

Efter halva projektperioden ska en fungerande simulering av CPU:n med vissa implementerade instruktioner kunna visas. Vi hoppas också kunna visa en tile-baserad bild på monitorn.

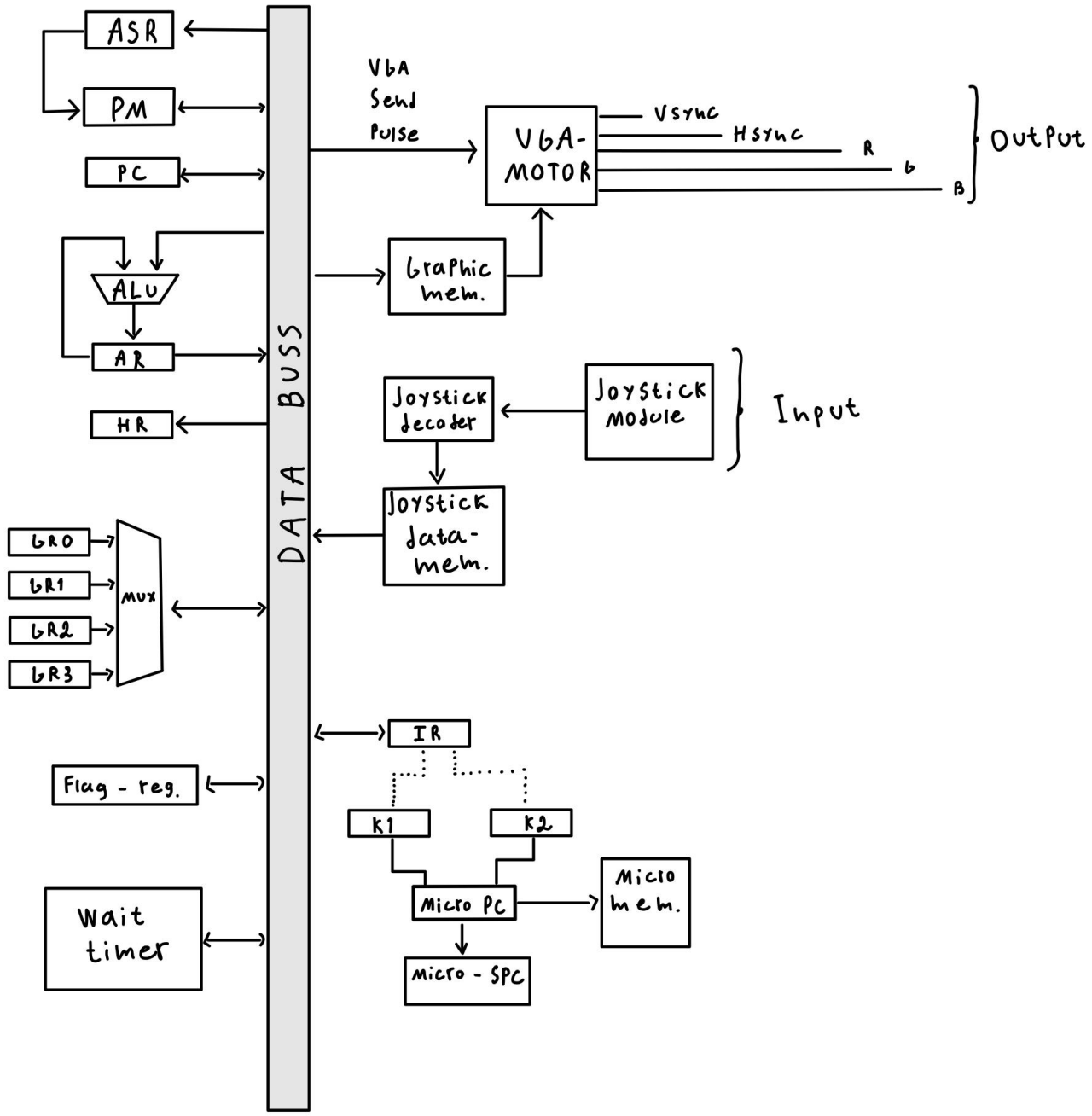
Blockschema

Se figur 2.

Mnemonic	**Funktion**
ADD. Rd, Ra	Rd <= Rd+Ra, uppdatera flag
ADDI. Rd, const	Rd <= Rd+const, uppdatera flag
AND. Rd, Ra	Rd <= Rd AND Ra, uppdatera flag
ANDI. Rd, const	Rd <= Rd AND const, uppdatera flag
BEQ. offset	Hopp om Z = 1
BGE. offset	Hopp om N $\oplus$ V = 0
BLT. offset	Hopp om N $\oplus$ V = 1
BNE. offset	Hopp om Z = 0
BMI. offset	Hopp om N = 1
BPL. offset	Hopp om N = 0
CMP. Rd, Ra	Rd - Ra uppd. flag
CMPI. Rd, const	Rd-const (Z,N,C,V)
COPY. Rd, Ra	Rd <= Ra
IRET	Återhopp från avbrott
JSR. offset	Subr.anrop till PC+1+offset
LD. Rd, Ra	Rd <= MEM(Ra)
LDI. Rd, const	Rd <= const
NOP	No operation
OR. Rd, Ra	Rd <= Rd OR Ra (Z, N, C, V)
ORI. Rd, const	Rd <= Rd OR const (Z, N, C, V)
POP. Rd	Rd <= DM(SP+1)
PUSH. Ra	DM(SP) <=Ra
RET	Återhopp från subrutin

RJMP	Hopp till PC+1+offset
ST	MEM(Rd) <= Ra

Figur 1, tillåtna instruktioner



Figur 2, blockschema av processorn