Datablad

Allmänt

- Processorns **pipeline** har **5 steg**.
- Naturlig ordlängd är **32 bitar**.
- Processorn har **separat data- och programminne**.
- Processorn har en **stack** (lagrad i dataminnet) som man modifierar med instruktionerna `PUSH` och `POP`.

Register

- Programräknaren (`PC`)
- Stackpekaren (`SP`)
- Flaggregistret (`FLAGS`)
- 29 generella register: `R0` till `R28`.

Flaggor

Flaggorna nås genom flaggregistret, som är ett specialregister i registerfilen.

Flagga	Beskrivning
`Z (Zero)`	Satt då resultatet av en ALU-instruktion blev lika med noll.
`N (Negative)`	Satt då resultatet av en ALU-instruktion är negativt. Endast intressant när man jobbar med tvåkomplement.
`V (Overflow)`	Satt då resultatet av en ALU-instruktion resulterade i overflow.
`C (Carry)`	Satt då resultatet av en ALU-instruktion resulterade i carry.

I/O

Processorn har minnesmappat I/O. Detta innebär att man kommunicerar med kringenheter genom att använda vanliga load/store-instruktioner, fast man använder till speciella adresser. Samtliga adresser går att både läsa och skriva till, men det primära syftet beskrivs i tabellen, under I/O.

Notera: Alla adresser är 0x00000000 i tabellen eftersom de kommer specificeras senare.

Address	I/O	Beskrivning
`0x00000000`	`OUT`	UART:ens konfiguration
`0x00000000`	`IN`	UART (tillstånd + data)
`0x00000000`	`IN`	Tangentbord (tillstånd + data)
`0x00000000`	`OUT`	Lampor (tillstånd)
`0x00000000`	`OUT`	Ljudenhetens konfiguration (bpm).
`0x00000000`	`OUT`	Ljudenhetens konfiguration (kanal 0).
`0x00000000`	`OUT`	Ljudenhetens konfiguration (kanal 1).
`0x00000000`	`OUT`	Ljudenhetens konfiguration (kanal 2).
`0x00000000`	`OUT`	Ljudenhetens konfiguration (kanal 3).
`0x00000000`	`IN`	Ljudenhetens tillstånd (kanal 0).
`0x00000000`	`IN`	Ljudenhetens tillstånd (kanal 1).
`0x00000000`	`IN`	Ljudenhetens tillstånd (kanal 2).
`0x00000000`	`IN`	Ljudenhetens tillstånd (kanal 3).
`0x00000000`	`OUT`	Kommunikation med grafikenheten
`...`	`...`	...
`0x00000000`	`OUT`	Kommunikation med grafikenheten

Instruktioner

Notera: Alla OP-koder är 000000 eftersom de kommer specificeras senare.

Notera: Instruktionsuppsättningen är preliminär och kommer förändras under projektets gång.
Det kan också hända att vissa instruktioner prioriteras bort eller (i de fall det är möjligt) implementeras som pseudoinstruktioner i assemblatorn istället.

Ladda/Spara

Load (LD)

Ladda in från dataminnet till det angivna registret.
Operation: `rD = data(rA + ext(offset))`
OP-kod: `000001`
Flaggor: -

6	5	5	16
OP	**D**	**A**	**offset**

Store (STR)

Skriv det angivna registret till dataminnet.
Operation: `data(rD + offset) = rA`
OP-kod: `000010`
Flaggor: -

6	5	5	16
OP	**D**	**A**	**offset**

Move (MOV)

Flytta data mellan två register.
Operation: `rD = rA`
OP-kod: `000011`
Flaggor: -

6	5	5	16
OP	**D**	**A**	**-**

Move High Immediate (MOVHI)

Ladda in immediate till de höga bitarna i det angivna registret.
Operation: `rD[0...15] = Immediate`
OP-kod: `000100`
Flaggor: -

6	5	5	16
OP	**D**	**-**	**I**

Move Low Immediate (MOVLI)

Ladda in immediate till de låga bitarna i det angivna registret.
Operation: `rD[16...31] = Immediate`
OP-kod: `000101`
Flaggor: -

6	5	5	16
OP	**D**	**-**	**I**

Push (PUSH)

Pusha det givna registret till stacken. Öka stackpekaren.
Operation: `SP += 4, data(SP) = rA`
OP-kod: `000110`
Flaggor: -

6	5	21
OP	**A**	**-**

Pop (POP)

Poppa toppen av stacken och spara det givna registret. Minska stackpekaren.
Operation: `rA = data(SP), SP -= 4`
OP-kod: `000111`
Flaggor: -

6	5	21
OP	**D**	**-**

Aritmetik

6	5	21
OP	**A**	**address**

Add (ADD)

Addera två register och spara det i ett tredje.
Operation: `rD = rA + rB`
OP-kod: `001010`
Flaggor: N, Z, V, C

6	5	5	5	11
OP	**D**	**A**	**B**	**-**

Add Immediate (ADDI)

Addera ett register med en konstant och spara det i ett annat register.
Operation: `rD = rA + ext(I)`
OP-kod: `001011`
Flaggor: N, Z, V, C

6	5	5	16
OP	**D**	**A**	**I**

Subtract (SUB)

Subtrahera ett register från ett annat, och spara det i ett tredje.
Operation: `rD = rA - rB`
OP-kod: `001100`
Flaggor: N, Z, V, C

6	5	5	5	11
OP	**D**	**A**	**B**	**-**

Subtract Immediate (SUBI)

Addera ett register med en konstant och spara det i ett annat register.
Operation: `rD = rA - ext(I)`
OP-kod: `001101`
Flaggor: N, Z, V, C

6	5	5	16
OP	**D**	**A**	**I**

Multiplication (MUL)

Multiplicera ett register med ett annat, och spara det i ett tredje.
Operation: `rD = rA * rB`
OP-kod: `001110`
Flaggor: N, Z, V, C

6	5	5	5	11
OP	**D**	**A**	**B**	**-**

Multiplication with Immediate (MULI)

Multiplicera ett register med en konstant, och spara det i ett tredje.
Operation: `rD = rA * ext(I)`
OP-kod: `001111`
Flaggor: N, Z, V, C

6	5	5	16
OP	**D**	**A**	**I**

Increase (INC)

Öka värdet i det givna registret med 1.
Operation: `rA = rA + 1`
OP-kod: `010000`
Flaggor: N, Z, V, C

6	5	21
OP	**A**	**-**

Decrease (DEC)

Minska värdet i det givna registret med 1.
Operation: `rA = rA - 1`
OP-kod: `010001`
Flaggor: N, Z, V, C

6	5	21
OP	**A**	**-**

Skift

Logic Shift Left (LSL)

Logiskt skifta källregistret vänster ett steg och placera i destinationsregistret.
Operation: `rD = rA << 1`
OP-kod: `010100`
Flaggor: -

6	5	5	16
OP	**D**	**A**	**-**

Arithmetic Shift Left (ASL)

Aritmetiskt skifta källregistret vänster ett steg och placera i destinationsregistret.
Operation: `rD = rA << 1`
OP-kod: `010101`
Flaggor: -

6	5	5	16
OP	**D**	**A**	**-**

Logic Shift Right (LSR)

Logiskt skifta källregistret höger ett steg och placera i destinationsregistret.
Operation: `rD = rA >> 1`
OP-kod: `010110`
Flaggor: -

6	5	5	16
OP	**D**	**A**	**-**

Arithmetic Shift Right (ASR)

Aritmetiskt skifta källregistret höger ett steg och placera i destinationsregistret.
Operation: `rD = rA >> 1`
OP-kod: `010111`
Flaggor: -

Logik

6	5	5	16
OP	**D**	**A**	**-**

And (AND)

"Ocha" de givna registerna och sparar resultatet i destinationsregistret.
Operation: `rD = rA & rB`
OP-kod: `011010`
Flaggor: -

6	5	5	5	11
OP	**D**	**A**	**B**	**-**

And Immediate (ANDI)

"Ocha" det givna registret med en omedelbar konstant och sparar resultatet i destinationsregistret.
Operation: `rD = rA & ext(I)`
OP-kod: `011011`
Flaggor: -

6	5	5	16
OP	**D**	**A**	**I**

Or (OR)

"Ora" de givna registerna och sparar resultatet i destinationsregistret.
Operation: `rD = rA | rB`
OP-kod: `011100`
Flaggor: -

6	5	5	5	11
OP	**D**	**A**	**B**	**-**

Or Immediate (ORI)

"Ora" det givna registret med en omedelbar konstant och sparar resultatet i destinationsregistret.
Operation: `rD = rA | ext(I)`
OP-kod: `011101`
Flaggor: -

6	5	5	16
OP	**D**	**A**	**I**

Not (NOT)

Inverterar det givna registret och sparar resultatet i destinationsregistret.
Operation: `rD = !rA`
OP-kod: `011110`
Flaggor: -

6	5	5	16
OP	**D**	**A**	**-**

Exclusive or (XOR)

"Exklusivt ochar" de givna registerna och sparar resultatet i destinationsregistret.
Operation: `rD`
OP-kod: `011111`
Flaggor: -

6	5	5	5	11
OP	**D**	**A**	**B**	**-**

Jämföra

Compare (CMP)

Jämför de givna registrerna och uppdatera flaggorna enligt resultatet.
Operation: `rA - rB`
OP-kod: `100011`
Flaggor: Z, N, V

6	5	5	16
OP	**A**	**B**	**-**

Compare Immediate (CMPI)

Jämför det registret och konstanten, uppdatera flaggorna enligt resultatet.
Operation: `rA - exit(I)`
OP-kod: `100100`
Flaggor: Z, N, V

6	5	21
OP	**A**	**I**

Hopp

Branch Equal (BEQ)

Hoppa om jämförelsen gav likhet.
Operation: `Z == 1 ? JMP : NOP`
OP-kod: `101000`
Flaggor: -

6	26
OP	**I**

Branch Not Equal (BNEQ)

Hoppa om jämförelsen inte gav likhet.
Operation: `Z == 0 ? JMP : NOP`
OP-kod: `101001`
Flaggor: -

6	26
OP	**I**

Branch Less Than (BLT)

Hoppa om jämförelsen gav att det ena registret var mindre än det andra.
Operation: `N != V ? JMP : NOP`
OP-kod: `101010`
Flaggor: -

6	26
OP	**I**

Branch Greater Than (BGT)

Hoppa om jämförelsen gav att det ena registret var större än det andra.
Operation: `(Z == 0) & (N == V) ? JMP : NOP`
OP-kod: `101011`
Flaggor: -

6	26
OP	**I**

Branch Less Than Or Equal (BLTEQ)

Hoppa om jämförelsen gav att det ena registret var mindre än eller lika med det andra.
Operation: `(Z == 1) || (N != V) ? JMP : NOP`
OP-kod: `101100`
Flaggor: -

6	26
OP	**I**

Branch Greater Than Or Equal (BGTEQ)

Hoppa om jämförelsen gav att det ena registret är större än eller lika med det andra.
Operation: `N == V ? JMP : NOP`
OP-kod: `101101`
Flaggor: -

6	26
OP	**I**

Jump (JMP)

Ovillkorigt hopp med offseten i det givna registret, och med en frivillig immediate offset.
Operation: `PC = PC + I + 1`
OP-kod: `101110`
Flaggor: -

6	26
OP	**I**