Laboratory 4 tasks

**Task 1**

*• Explain how each of the ALU functions are defined. In particular, you need to be able to*

*explain how subtraction works, including the use of two's complement.*

Sub, control MSB is 1 -> carry in

1st bit = 1, B gets inverted

A, inverted B + carry in goes into adder, since 2’s complement is inverting a number and adding a one.

SLT

Eftersom vi gör en subtraktion när vi kör SLT ( A -B) så kommer vi få en MSB som är 1 eller 0 beroende på vilket tal som var större- Då vi har 2’s komplement så kommer en 1’a att signalera att B är större (Alltså svaret blir ett negativt tal då MSB är signbit). Denna 1a shiftar vi till LSB med en shifter med shift constant på 31(0x1f).

*• How did you implement the logic for the Zero output port? Did you consider any*

*alternatives? Be prepared to explain your design choices.*

<http://www.cburch.com/logisim/docs/2.3.0/libs/arith/comparator.html>

The "zero" output tells us if the alu result is zero or not: zero=1 means the output was zero, zero=0 means the output is not zero.

East edge, labeled = (output, bit width 1)

1 if the first input equals the second input, 0 if the first input is not equal the second input.

Kör comparator equal med en constant (zero) För att kolla om output från ALUn är noll eller inte. Om den är noll kommer comparator att skicka ut en 1a för att signalera att resultatet av ALU operationen var noll. Orsaken till att vi använder comparator är av ren bekvämlighet. Den fanns redan som en implementerad komponent och den passade till syftet.

Man kunde absolut ha implementerat detta med

*• What is the purpose of the ALU? Why are several functions grouped together into one*

*component?*

ALU är den mest centrala componenten I en dator då det är den som faktiskt gör alla beräkningar som krävs för att datorn ska kunna köra kod. Funktionerna är tillsammans för att det ska gå snabbare att göra beräkningar??

**Task 2**

*• Explain if the read operation or the write operation, or both operations are clocked (updated*

*at the clock edge). Why is it implement this way?*

*“South edge, indicated with a triangle (input, bit width 1)*

*Clock input: At the instant that this input value rises from 0 to 1 (the rising edge), the register's value will be updated to the value of the D input.”*

Write är styrd av klockpulsen på rising edge.

READ???

*“A register stores a single multi-bit value, which is displayed in hexadecimal within its rectangle, and is emitted on its Q output. When the clock input (indicated by a triangle on the south edge) indicates so, the value stored in the register changes to the value of the D input at that instant. Exactly when the clock input indicates for this to happen is configured via the Trigger attribute.”*

*• Explain the semantics of reading from and writing to $0, and how you implemented this*

*behavior.*

WE3(Write enable) och dekodade register id 000 säger till att vi vill skriva till registret. Den data som lagras där är från en konstant som vi har definierat och inte från själva datan som skickas in i registrena.

*• How many bits of data can this register file store? If the address width was the same size as*

*for a complete 32-bits MIPS processor, how many bits would in such a case such register*

*file store?*

?

7x32=224?

**Task 3**

*• Explain how you have implemented the control signals for the beq instruction. Why is this a*

*correct solution?*

OP-code for beq is 000100 (4), so we need another comparator that compares opcode with constant 4.

Vi måste lägga till en mux för att vid en branch så om condition inte är mött, då skickas vi helt enkelt vidare till nästa instruktion så PC ska bara inkrementeras med 4. Om det sker en branch så måste det PC ökas med Immediate.

Beq fungerar så att man jämför två register, detta görs med en sub (0x6 eller 110 ). Detta för att det är lättare att jämföra två tal med sub än add, då om svaret är 0 så är de lika.

Eftersom vi egentligen bara har 1 bit som ALUop så blir därför den andra i MUXen add. Tittar man på föreläsning 9 så¨ser man även att alla ALUop slutar på noll, förutom beq och j (som är ??)

*• Be prepared to explain why the RegDst control signal or the AluSrc signal is hooked up to*

*certain signals. You should be prepared to explain this using the following figure.*

RegDst determines how the destination register is specified (rt = 0 or rd = 1). Only applicable on R-type.

AluSrc selects the second source operand for the ALU (rt or sign-extended immediate field in Patterson and Hennessey).

**Task 4**

*• Explain how the bit selection works for the alternatives that are controlled by the RegDst*

*control signal. Which instructions are using what logic and why?*

RegDst kontrollerar mux. Den väljer om destinationsregistret ska va rd eller rt.

????

*• Explain how the beq instruction is implement, how the address is calculated, and how the*

*signals are controlled by the control unit.*

Branch instruction ger 1 från Control. Samtidigt skickas även 110 (6) som ALUcontrol vilket är en Sub. Med Sub så kan man jämföra värdet i två register genom att se på Zero flag i ALUn. Är den 1 så är registrena lika.

Beq är en immidiate funktion, och det som står i immidiate fältet skickas till sign extender.

Immidiate fältet innehåller egentligen en offset som säger till hur många instruktioner i PC som branchen måste ”passera”. Den kan vara positiv eller negativ.

PC + 4 + (Signed immediate)\*4 | Här kan signed immediate va negativ

Shiftas sedan med 2 då det är samma sak som att multiplicera något med 4.

Sedan adderas den med PC + 4 som är definierat i början av kretsen.

**Task 5**

*• Show and explain how the factorial function works for arbitrary input value n (the teaching*

*assistant will give you the value that you should test). Be prepared so that you know how to*

*change the input value easily.*

GG EEEEEZ

*• Explain how you implemented unconditional jumps in your program.*