Laboration 4

Uppgift 1

Explain how each of the ALU functions are defined. In particular, you need to be able to explain how subtraction works, incuding the use of two's complement.

För subtraction tas MSB som carry in som är 1 vid F = 110. LSB är 0, B blir inverterad. Så A, ~B och carry går in i addern.

Two's complement där binära siffran inverteras och 1 adderas. Eftersom en subtraktion görs kommer MSB bli 1 eller 0 beroende på vilket tal som är störst. Med two's complement betyder en 1'a negativt tal och därför är B större. Denna 1'a shiftas till LSB med en shifter med konstanten 0x1f, 32 bitar data.

How did you implement the logic for the Zero output port? Did you consider any alternatives? Be prepared to explain your design choices.

Zero out visar om ALUns resultat är noll eller inte. Så om noll är output = 1. Därför om inte noll är output = 0.

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

Sedan en comparator equal med konstanten 0 för att se om output från ALUn är noll. Om noll kommer comparatorn skicka en 1a. Varför? För att den passade och fungerade smidigt.

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

En ALU utför enkla aritmetiska beräkningar som multiplikation, division, subtraktion, addition. Det är den mest centrala komponenten i en dator för utan den kan inte kod köras. Jag antar att flera funktioner är i ALU'n eftersom att det ska bli snabbare att utföra dessa beräkningar eftersom det då skickas värden till samma komponent, istället för att ha olika komponenter för olika slags operationer.

Uppgift 2 *

Explain if the read operation or the write operation, or both operations are clocked (updated at the clock edge). *Why is it implemented this way?*
http://www.cburch.com/logisim/docs/2.7/en/html/libs/mem/register.html

"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."

Write är styrd av klockan på rising edge.
Read, vet inte. *

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

WE3 (write enable) och decodeade register vid 000 säger till att vi vill skriva till registret. Den data som lagras där är från en konstant som vi definierat och inte från själva datat som skickas in till 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?

224. 32*7=224.

Uppgift 3 *

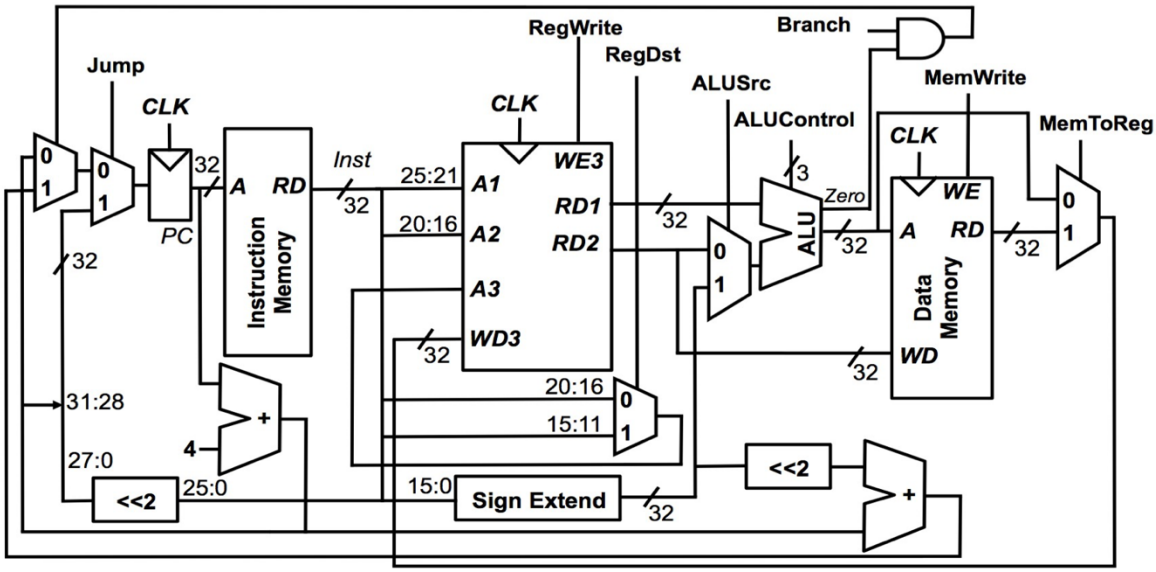
Explain how you have implemented the control signals for the beq instruction. Why is this a correct solution?

OP-koden för beq är 000100 (4). Därför behövs en comparator som jämför opcode med konstanten 4. Vi lägger till en mux eftersom man skickas vidare till nästa instruktion när en condition inte blir mött vid en branch, så PC ska bara inkrementeras med 4. Om en branch sker så måste PC ökas med immediate.

Beq jämför två register, vilket 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 den andra i MUXen add. På föreläsning 9 ses även att ALUop slutar på noll, men inte beq och j.

Be prepared to explain why the RedDst control signal or the AluSrc signal is hooked up to certain signals. You should be prepared to explain this using the following figure.



RegDst avgör hur destinationsregistret är specifierad (rt = 0 eller rd = 1). Gäller endast R-type. AluSrc väljer ”second source operand” för ALUn. * (rt eller sign-extended immediate field i P&H).

Uppgift 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 controllerar mux, den väljer om destinationsregistret ska vara rd eller rt.

Explain how the beq instruction is implemented, how the address is calculated, and how the signals are controlled by the control unit.

Beq (branch if not equal) instruktionen ger 1 från Control. Samtidigt skickas även 110 (6) som ALUcontrol vilket är sub. Med sub kan värdet i två register jämföras genom att se på Zero flag i ALUn. Är den 1 så är registrena lika. Beq är en immidiate funktio, och det som står i immidiate fältet skickas till sign extender.

Immidiate fältet innehåller egentligen en offset som ser till hur många instruktioner i PC som branschen måste ”passera”. Kan vara positiv eller negativ. PC + 4 + (signed immidiate)*4 , här kan signed immidiate vara negativ. Shiftas sedan med 2, det är samma sak som multiplikation med 4. Sedan adderas den med PC + 4 som är definierat i början av kretsen.

Uppgift 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.

Explain how you implemented unconditional jumps in your program.