CTA01 – Huiswerk week 5

Vandaag hebben we paragraaf 4.1 t/m 4.5 van het boek behandeld. De slides staan op de de wiki in pptx en pdf formaat. We hebben van H4 slide 1 t/m 49 behandeld.

Huiswerk

Lees paragraaf 4.1 t/m 4.5 door. Vragen over de opgaven graag uiterlijk volgende lesweek dinsdag 23:59 uur mailen zodat ik ze op woensdag in de les kan behandelen. Vergeet niet om de 'check yourself' oefeningen te maken. De antwoorden staan op p. 384.

Maak de volgende opgaven uit paragraaf 4.17: 4.1 t/m 4.7, 4.14, 4.16 t/m 4.18 en 4.20

Vragen over de opgaven graag uiterlijk volgende lesweek dinsdag 23:59 uur mailen zodat ik ze op woensdag in de les kan behandelen. Maak een issue aan op de wiki als je een fout in dit document hebt gevonden.

Mijn antwoorden

4.1.1 `RegWrite` = `1`
`ALUSrc` = `0`
`ALUoperation` = `AND` = `0000`
`MemWrite` = `0`
`MemRead` = `0`
`MemToReg` = `0`

4.1.2 Registers, `ALUsrc` mux, ALU, and the `MemToReg` mux

4.1.3 Data Memory produces no output. The output of the `Signextend` and the `Zero` output of the ALU are not used.

4.2 `Reg2Loc` for **`LDUR`**: When executing **`LDUR`**, it doesn't matter which value is passed to `Read register 2`, because the `ALUSrc` mux ignores the result-

ing Read data 2 output and selects the sign extended immediate value instead.

MemToReg for **STUR** and **CBZ**: Neither **STUR** nor **CBZ** write a value to the register file. It doesn't matter which value the MemToReg mux passes to the register file because the register file ignores that value.

4.3.1 35 %

4.3.2 100 %

4.3.3 48 %

4.3.4 The sign extend produces an output during every cycle. If its output is not needed, it is simply ignored.

4.4.1 Only **LDUR** is broken.

4.4.2 I-type, **LDUR**, and **STUR** are all broken.

4.4.3 **STUR**, **CBZ**, and **CBNZ** are all broken.

4.5 For context: The encoded instruction is **STUR** X2, [X3, #0x14]

4.5.1 Sign extend = 0x14
Shift left 2 = 0x50

4.5.2 ALUOp = 00, ALU control = Add = 0010

4.5.3 The new PC is the old PC + 4. This signal goes from the PC, through the PC+4 adder, through the branch mux, and back to the PC.

4.5.4 Reg2Loc: inputs 00001 and 00010. Output 00010
ALUSrc: inputs 0x0000000000000014, Reg[X2]. Output 0x0000000000000014.
Memtoreg: inputs Reg[X3]+0x0000000000000014, undefined. Output can be any of the inputs (don't care).
PC: PC + 4, PC + 0x0000000000000050. Output PC + 4.

4.5.5 ALU inputs: Reg[X3], 0x0000000000000014
PC+4 adder inputs: PC, 4.
Branch adder inputs: PC, 0x0000000000000050.

4.5.6 Read register 1: 3
Read register 2: 2

Write register: 2
Write data: output of MemtoReg mux (don't care).

4.6.1 No additional logic blocks are needed. But the signextend block must pass the 12 bits ALU_immeadiate field **without** sign extention.

4.6.2 Reg2Loc: X (don't care) The value of Reg2Loc is a don't care because the ALUSrc mux ignores the resulting Read data 2 output.
Uncondbranch: 0
Branch: 0
MemRead: 0
MemToReg: 0
ALUop: 10, ALU control: 0010
MemWrtie: 0
ALUsrc: 1
RegWrite: 1

4.7.1 Figure 2.20 shows that instruction bit 28 contains the correct value for the Reg2Loc control line. In general, the value of the Reg2Loc wire is 0 for R-type instructions, 1 for D-type and CB-type instructions, and don't care for everything else. Reg2Loc is also a don't care for **LSL** and **LSR** because, even though they use an R-type format, they don't use a second register operand.

4.7.2 725 ps

4.7.3 925 ps

4.7.4 880 ps

4.7.5 730 ps

4.7.6 525 ps

4.7.7 675 ps

4.7.8 925 ps

4.14 Although many instructions use the sign-extend block, it is on the critical path of the **B** instruction only. All other instructions that use sign extend also use the register file, which is slower.

4.16.1 Pipelined: 350; non-pipelined: 1250

4.16.2 Pipelined: 1750; non-pipelined: 1250

4.16.3 Split the ID stage. This reduces the clock-cycle time to 300 ps.

4.16.4 35 %

4.16.5 65 %

4.17 Let's look at when each instruction is in the WB stage. In a k-stage pipeline, the first instruction doesn't enter the WB stage until cycle k. From that point on, at most one of the remaining $n-1$ instructions is in the WB stage during every cycle. This gives us a minimum of $k+(n-1)$.

4.18 X3 = 33 and X4 = 26

4.20

```
ADDI    X1, X2, #5
NOP
NOP
ADD     X3, X1, X2
ADDI    X4, X1, #15
NOP
ADD     X5, X3, X2
```