Facultad de Informática
Universidad Complutense de Madrid

INTRODUCTION TO COMPUTERS II
MODULE 5

Basic problems:

1. Assume that the instructions of a program running in a single-cycle RISC-V follow this distribution:

	add-like	**addi**-like	**lw**	**sw**	**beq**	**jal**
frequency	24%	28%	25%	10%	11%	2%

Provide:

- The percentage of instructions that use the data memory.
- The percentage of instructions that use the instruction memory.
- The percentage of instructions that use the sign extension module.
- The percentage of instructions that use the ALU.

2. Provide the value of the control signals produced in a single-cycle RISC-V when executing a **lw** instruction.

3. Provide the value of the control signals produced in a single-cycle RISC-V when executing an **andi** instruction.

4. Provide the value of the control signals produced in a single-cycle RISC-V when executing a **beq** instruction.

5. Suppose that a single-cycle RISC-V reads the 0x00c6aa23 word from the 0x00001000 instruction memory address. Also, assume that all registers contain a value equivalent to their name number, i.e., x1 contains 0x00000001, x2 contains 0x00000002, etc. Also, all data memory positions contain 0x00000000. Provide:

- The value of the ALUctr signal generated by the ALU DEC.
- The value of the ImmSrc signal generated by the sExt DEC.
- The new value of the PC after the execution of the given instruction.
- The values of the data inputs and outputs of the multiplexers.
- The values of the register file outputs.
- The values at the inputs of the ALU and the adders.

Additional problems:

6. Indicate what instruction is being executed on the single-cycle processor, if the values of the control signals are the following:

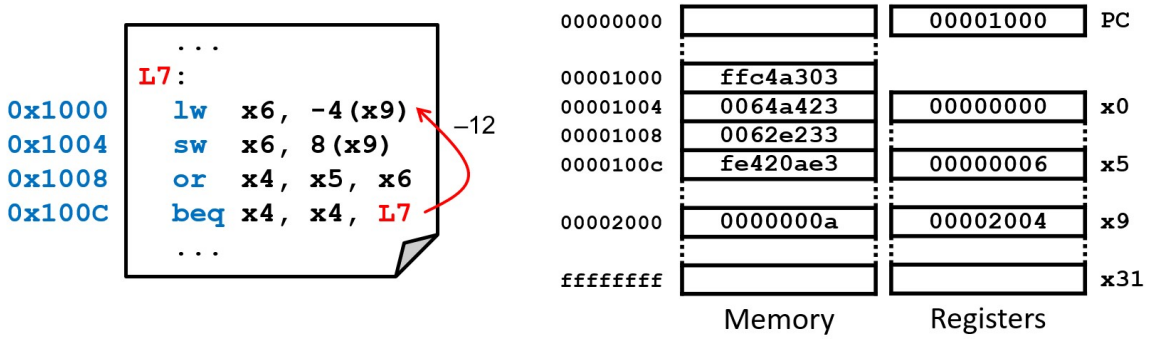
a) MemWr = 1
b) ALUSrc = 0 and ALUctr = 010
c) BRwr = 0 and ALUctr = 001
d) PCsrc = 1 and Jump = 1
e) ALUSrc = 1 and ALUctr = 000
f) ResSrc = 10
g) ResSrc = 00
h) Branch = 1

7. When silicon chips are fabricated, defects in materials and manufacturing errors can result in defective circuits. A very common defect is for one signal wire to get damaged and always register a logical 0 (stuck-at-0) or a logical 1 (stuck-at-1). Indicate what single-cycle RISC-V instructions would fail if the following control signals suffered a stuck-at-0 defect:

a) BRWr
b) $ALUop_1$
c) $ALUop_0$
d) MemWr
e) $ImmSrc_1$
f) $ImmSrc_0$
g) $ResSrc_1$
h) $ResSrc_0$
i) PCSrc
j) ALUSrc

8. Repeat the previous exercise, but now with a stuck-at-1 defect.

9. Assume that the following program is running on a single-cycle RISC-V, with the initial value of the memory and registers shown in the picture. Represent the execution diagrams for the registers and the memory, as well as for the control and status signals, so that their values are shown for each clock cycle.



10. Discuss the modifications required in the data path of the reduced architecture single-cycle RISC-V, in order to extend its ISA with each of the following instructions:

a) **`xor` / `xori`**
b) **`sll` / `slli`**
c) **`bne` / `blt` / `bge`**
d) **`slt` / `slti`**
e) **`lb` / `lh` / `lbu` / `lhu`**
f) **`sb` / `sh`**
g) **`jarl`**
h) **`lui`**
i) **`auipc`**

11. Discuss the modifications required in the data path of the reduced architecture single-cycle RISC-V, in order to extend its ISA with a new I-type instruction that reads a word from memory, pre-incrementing the base register, **`lwpreinc` *`rd, imm(rs1)`***:

$$\{ \text{rd} \leftarrow \text{Mem}[\ \text{rs1} + \text{sExt(imm)}\ ],\ \text{rs1} \leftarrow \text{rs1} + \text{sExt(imm)}\ \}$$

12. Discuss the modifications required in the data path of the reduced architecture single-cycle RISC-V, in order to extend its ISA with a new I-type instruction that reads a word from memory, post-incrementing the base register, **`lwpostinc` *`rd, imm(rs1)`***:

$$\{ \text{rd} \leftarrow \text{Mem}[\ \text{rs1}\ ],\ \text{rs1} \leftarrow \text{rs1} + \text{sExt(imm)}\ \}$$

13. Discuss the modifications required in the data path of the reduced architecture single-cycle RISC-V, in order to extend its ISA with a new R-type instruction that swaps the value of 2 registers, **`swap` *`rs1, rs2`***:

$$\{ \text{rs1} \leftarrow \text{rs2},\ \text{rs2} \leftarrow \text{rs1}\ \}$$

14. Discuss the modifications required in the data path of the reduced architecture single-

cycle RISC-V, in order to extend its ISA with a new U-type instruction that loads a data read from a certain memory position into a register, **`lwa rd, imm`**:

{ rd ← Mem[zExt(imm)] }

15. If you could reduce the delay of just one of the components of the single-cycle RISC-V in the 90nm CMOS library by half, which one would you choose to get a smaller cycle time? What would be the value of the new cycle time?

16. If the ALU delay is reduced a 30%, the extension sign module delay a 10% and the register file read/write delays a 20% (referred to the single-cycle RISC-V with the 90nm CMOS library), calculate the critical path delay of each instruction and how much time this processor would take to execute 100 million instructions.

17. Suppose that adding a multiplication instruction to the single-cycle RISC-V ALU doubles its delay, but this reduces the count of executed instructions a 5% (because multiplications do not have to be emulated by SW now).

a) Calculate the clock frequency of the new processor.
b) Discuss whether this change makes sense.
c) Calculate the maximum delay increment for the ALU so that this change keeps making sense.
d) Calculate the minimum percentage of the instruction count reduction so that this change makes sense.

18. The **`lw`** instruction is the slowest of the single-cycle RISC-V because it needs to use all the resources of the data path. If the behavior of the **`lw`** and **`sw`** instructions is modified so that the address that is sent to memory is the one stored in the rs1 base register (skipping the addition with the offset), the cycle time can be reduced. However, this forces to replace the old **`lw`** and **`sw`** instructions with **`lw`**/**`addi`** and **`sw`**/**`addi`** pairs, in all the programs. Assuming that the original processor executes a program with 10^8 instructions, following this distribution:

	`add`-like	**`addi`**-like	**`lw`**	**`sw`**	**`beq`**	**`jal`**
frequency	24%	28%	25%	10%	11%	2%

a) Calculate the total number of instructions that the new program will have.
b) Calculate the cycle time of the new processor.
c) Calculate the ratio between the execution times of the program running on both processors (speedup).
d) Discuss whether this change makes sense.

19. Suppose that doubling the number of general purpose registers, from 32 to 64, in the single-cycle RISC-V reduces the number of the executed **`lw`** and **`sw`** instructions a 12%. However, this increases the register file access delay a 25% and its cost a 50%. Assuming that the original processor executes a program with 10^8 instructions, following this distribution:

	`add`-like	**`addi`**-like	**`lw`**	**`sw`**	**`beq`**	**`jal`**
frequency	24%	28%	25%	10%	11%	2%

a) Calculate the total number of instructions that the new program will have.
b) Calculate the cost and the cycle time of the new processor.
c) Calculate the execution time ratio for both processors.

Solutions

1. a) 35% b) 100% c) 76% d) 98%

2. Branch = 0, Jump = 0, BRwr = 1, ALUsrc = 1, ALUop = 00, MemWr = 0
ResSrc = 00, ALUctr = 000, ImmSrc = 11

3. Branch = 0, Jump = 0, BRwr = 1, ALUsrc = 1, ALUop = 10, MemWr = 0
ResSrc = 01, ALUctr = 010, ImmSrc = 00

4. Branch = 1, Jump = 0, BRwr = 0, ALUsrc = 0, ALUop = 01, MemWr = 0
ResSrc = −, ALUctr = 001, ImmSrc = 10

5. a) ALUctr = 000 b) ImmSrc = 01 c) PC = 0x1004
d) Mux-PC: E_0 = 0x00001004, E_1 = 0x00001014, S = 0x00001004
Mux-ALU: E_0 = 0x0000000d, E_1 = 0x00000014, S = 0x00000014
Mux-WB: E_0 = 0x00000000, E_1 = 0x00000021, E_2 = 0x00001004, S = −
e) RD1 = 0x0000000d, RD2 = 0x0000000c
f) ALU_{up} = 0x0000000d, ALU_{down} = 0x00000014
Add_{PC} = 0x00001000
Add_{up} = 0x00001004, Add_{down} = 0x00001000

6.
a) **`sw`**
b) **`and`**
c) **`beq`**
d) **`jal`**
e) **`addi`**
f) **`jal`**
g) **`lw`**
h) **`beq`**

7.
a) **`lw`**, R/I-types, **`jal`**
b) R/I-types
c) **`beq`**
d) **`sw`**
e) **`jal`**, **`beq`**
f) **`sw`**, **`jal`**
g) **`jal`**
h) R/I-types
i) **`jal`**, **`beq`**
j) **`lw`**, **`sw`**, I-type

8.
a) **`sw`**, **`beq`**
b) **`lw`**, **`sw`**, **`beq`**
c) **`lw`**, **`sw`**, R/I-types
d) **`lw`**, **`beq`**, **`jal`**, R/I-types
e) **`lw`**, **`sw`**, I-type
f) **`lw`**, **`beq`**, I-type
g) **`lw`**, R/I-types
h) **`lw`**, **`jal`**
i) **`lw`**, **`sw`**, R/I-types
j) **`beq`**, R-type

9. See slides.

10. a) Extend the functionality of the ALU.
b) Extend the functionality of the ALU.
c) Extend the functionality of the ALU.
d) Extend the functionality of the ALU.
e) Add width selection lines to the Data Memory.
Add a sign/zero extender to the output of the Data Memory.
f) Add width selection lines to the Data Memory.
g) Connect the output of the ALU to a new input of the PC-MUX.
h) Extend the functionality of the Sign Extender.
Connect the output of the Sign Extender to a new input of the WB-MUX.
i) Extend the functionality of the Sign Extender.

11. Add another write data input to the Register File (RF) and connect it to the output of the ALU.
Add another write address input to the RF and connect it to bits 19:15 of the output of the Instruction Memory.
Add another write control input to the RF.

12. Add another write data input to the Register File (RF) and connect it to the output of the ALU.
Add another write address input to the RF and connect it to bits 19:15 of the output of the Instruction Memory.
Add another write control input to the RF.
Add a MUX to the address input of the Data Memory and connect its inputs to the output of the ALU and to the RD1 output of the RF.

13. Connect the RD2 output of the RF to a new input of the WB-MUX.
Connect a new MUX to the WA input with two inputs connected to bits 11:7 and 19:15 of the output of the Instruction Memory.
Add another write data input to the RF and connect it to the RD1 output of the RF.
Add another write address input to the RF connected to bits 24:20 of the output of the Instruction Memory.
Add another write control input to the RF.

14. Extend the Sign Extender.
Add a MUX to the address input of the Data Memory and connect its inputs to the output of the ALU and to the output of the Sign Extender.

15. Instruction memory, 23366 ps

16. **`lw`**: 24921 ps **`sw`**: 24061 ps I-type: 16421 ps R-type: 16134 ps
`beq`: 16421 ps **`jal`**: 9932 ps t_{clk} = 24921 ps t_{exec} = 2,49 s

17. a) 27,8 MHz b) no c) 1453 ps d) 23,2%

18. a) $135 \cdot 10^6$ b) 19116 ps c) 1,07 d) yes

19. a) $95{,}8 \cdot 10^6$ b) 27792 ps, 84883 μm^2 c) 1,04