German University in Cairo
Media Engineering and Technology
Assoc. Prof. Dr. Hassan Soubra

Computer System Architecture, Spring Semester 2022
Practice Assignment 7

Discussion: 07/05/2022 - 12/05/2021

Exercise 7-1

a) What is the longest (time wise) instruction in MIPS? Why?

b) What could be the second longest instruction in MIPS? Why?

c) Consider the following component times:

- Instruction Fetch (IF): 200ps
- Instruction Decode (ID): 100ps
- Execute (EX): 200ps
- Memory (MEM): 200ps
- Write Back (WB): 100ps

What is the frequency of the processor in case of a Single-Cycle (SC) implementation?

Solution:

a) Load Word (LW), it takes 5 functional units to execute.

b) Store Word (SW), it takes 4 functional units to execute, including memory, and usually memory is the slowest functional unit.

c) LW is the slowest so 200 +100+200+200+100=800ps.
$(1*10^{12})/800$ = 1.25 GHz

Exercise 7-2

a) Why is a sign extend module needed in the simple implementation of MIPS?
b) What MIPS instructions covered in class use the sign extend hardware module?

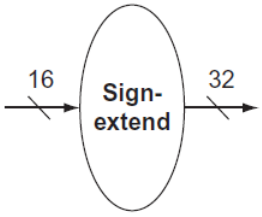
Solution:

a) To extend the 16-bit operand to 32 bits to use in a register for example.
b) LW, SW, BEQ, BNE, ADDI, etc...
- *You can know from the instruction description found in the MIPS Instruction Set Architecture document*

Imagine instructions with the following offsets:
c) 1000 1000 0000 0001
d) 0000 1000 0000 0001
What is the output of the sign-extend module?

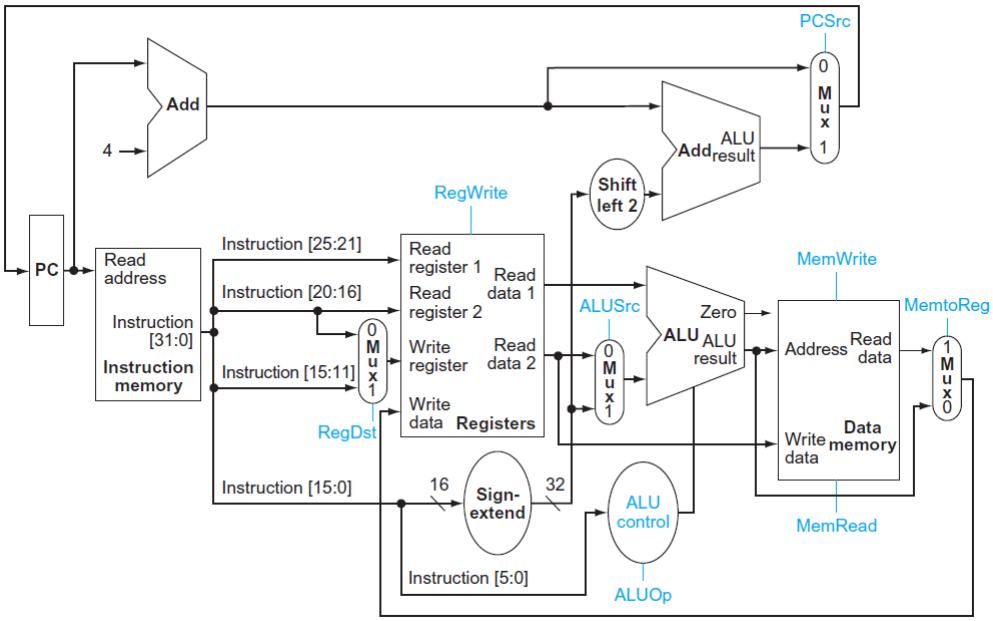
Solution:

c) 1111 1111 1111 1111 1000 1000 0000 0001
d) 0000 0000 0000 0000 0000 1000 0000 0001



Exercise 7-3

The following figure shows the MIPS datapath implementation we looked at in class. One of the missing instructions is the jump instruction. We want to extend the datapath to include the jump instruction.



1. What is the address field in the Jump instruction?
2. To what instruction seen in class the Jump looks like? What is the main difference between the two?
3. The effective target address is in the "current" 256 MebiBytes region. How is the destination address for a jump instruction formed? Hint: the low-order 2 bits of a jump address are always 00 (base two).

Solution:

1. The 26-bit field in jump instructions is a word address, meaning that it represents a 28-bit byte address.
2. Jump looks like a branch instruction but it is not conditional.
3. The destination address of a jump instruction is formed by concatenating the upper 4 bits of the current PC + 4 (the effective target address is in the "current" $2^{(32-4)} = 2^{28} = 256$ MB MebiBytes region) to the 26-bit address field in the jump instruction and adding 00 as the 2 low-order bits "multiply by 4 / shift left logical by 2" (byte addressing of a word always ends with 00).

Jump Address Calculation:

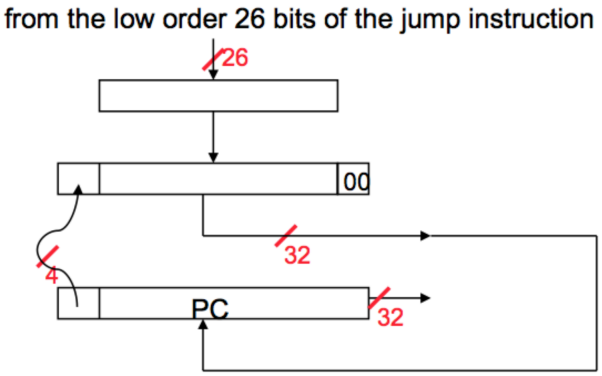


Figure 1: The transformation of the 26-bit address field to the 32-bit jump address

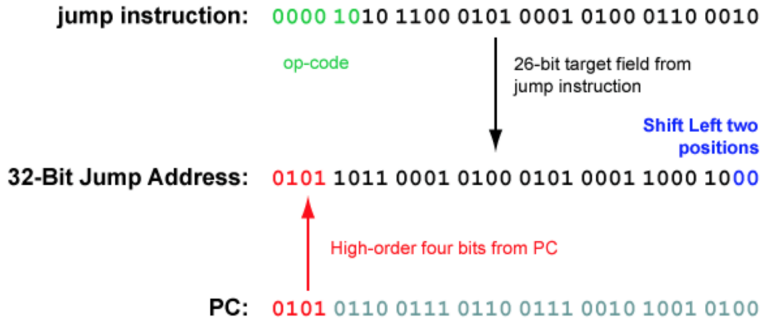


Figure 2: An example of creating the 32-bit jump address from the 26-bit address field

The MIPS Single-Cycle Datapath

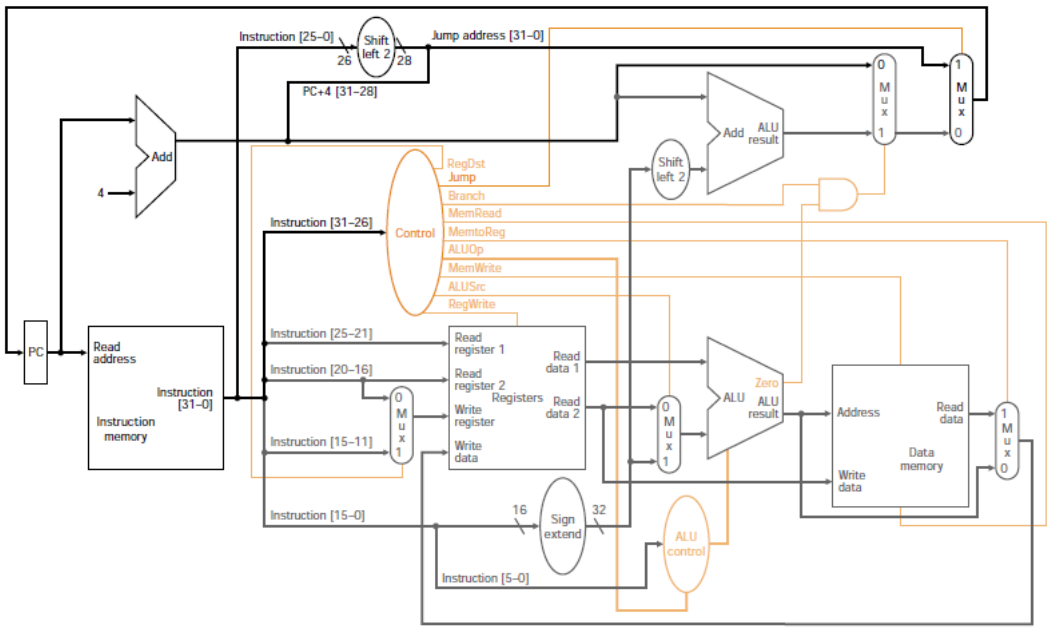
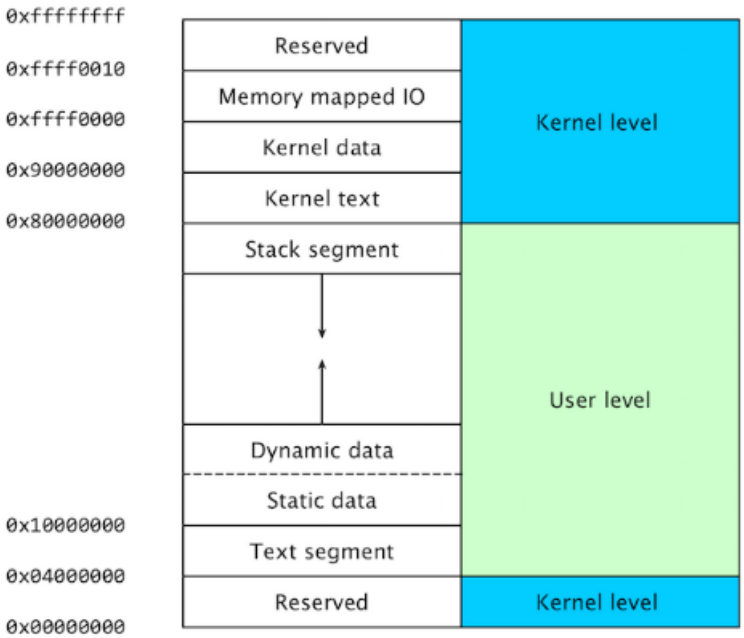


Figure 3: The MIPS datapath after adding the jump address calculation and the control unit

Exercise 7-4

Explain the different segments of the MIPS memory.



a) What is $gp? What memory segment does the $gp point to?

b) What is the size of the memory segment pointed by $gp? (The heap starts at 0x10010000)

c) $gp is initialized with a value so that it can rapidly access using offsets any place in the segment. What is the initial value?

Solution:

a) $gp = global pointer = MIPS register used conventionally to simplify access to static data.

b) Static data segment is from 0x10000000 to 0x1000FFFF, therefore the size = (0x1000FFFF - 0x10000000) + 1 = 65536 bytes = 64 KibiBytes.

c) Initial value = Middle of the static data segment (0x10008000).
Calculate the middle point offset: 65536 / 2 = 32768 = 0x00008000
0x10000000 (start of static data) + 0x00008000 (steps to reach middle point) = 0x10008000

We did not initialize the $gp with the start address of the static data (0x10000000) because the immediate field maximum value in the I-Format instructions is $(2^{16}) = 65536$ if the instruction uses it as an unsigned value. However, instructions like "LW" and "SW" use a signed immediate (offset) value. Therefore, we have 1 bit dedicated to the sign making the range (-2^{15} to $2^{15} - 1$).
Now, having the $gp pointing to the middle of the static data will allow us to move upward (positive) and downward (negative), covering the entire range of 65536 values (which is divided into half negative and half positive).