UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
COMPUTER SRTRUCTURE
Miniexam Examples

Exercise 1. There is a 20-bit processor with a register bank of 24 registers, which addresses a main memory at byte level. Answer correctly and briefly to the following questions:

a) What is the program counter and how many bits it stores?

It is the register whose content is the address of the next instruction to be executed. 20 bits.

b) What is the instruction register and how many bits does it store?

It is the record whose content is the instruction in progress. 20 bits.

c) What is the MBR register and how many bits does it store?

It is the register that contains the value brought/carried from/to memory. 20 bits.

d) If a register stores a value in one's complement, what range of values can it store?

$[-2^{20-1}+1 \ldots -0,0, \ldots 2^{20-1}-1]$

e) If a register stores a value in two complement, what range of values can it store?

$[-2^{20-1} \ldots -1,0, \ldots 2^{20-1}-1]$

f) f) How many bits are used to identify a record?

If there are 24 registers, with 5 bits (logarithm in base 2 for excess).

g) How many bits are stored in a register?

If it is a 20-bit word processor (registers are 20 bits) then 20 bits.

h) How many bits are used to address a memory cell?

If it is a 20 bits word processor (registers are 20 bits) then 20 bits.

i) How many bytes are stored in a memory location?

If it is a memory addressed at byte level then 1 byte.

Exercise 2. Add the following numbers represented in IEEE754 format, briefly indicating the general steps to be followed: 0x41900000 + 0x7F800000

0x41900000

0100 0001 1001 0000 0000 0000 0000 0000
0x 4 1 9 0 0 0 0 0

Sign = 0 (positiv)
Exponent = 10000011 (131 – 127 = 4)
Mantissa= 1001 (with implicit bit)

$1{,}001_2 \times 2^4 = 10010{,}0_2 \times 2^0 = 18_{10}$

0x7F800000

It is a special case: positive, exponent to ones and mantissa to zeros -> + infinity

As it is a special case, the result is infinite (0x7F800000)

Exercise 3. Given the following MIPS32 frgament code::

```
.data
        str1: .asciiz "hello word"
.text
            .glob main
main:
            li    $v0  0
            li    $t1, 0
            li    $t2  str1
if1:        lw    $t3  $t2
      beqz $t3, fin1
            add   $v0, $v0 1
            add   $t2  $t2 4
            b if1
fin1:       jr $ra
```

Indicate **all** the **errors** contained in the previous fragment, if you want to calculate the length of the character string and save it in the $v0 register.

The errors are the follwing:

```
            li    $v0  0
            li    $t1, 0
            la    $t2  str1
if1:        lb    $t3  ($t2)
      beqz $t3, fin1
            addi  $v0, $v0 1
            addi  $t2  $t2 1
            b if1
fin1:       jr $ra
```

Exercise 4. There is a processor with 32 registers of 64 bits each. The memory is addressed at byte level and the numbers are represented in complement to two. Answer correctly and briefly to the following questions:

a) ¿ How many GB of main memory can this computer address

You can address 2^{64} bytes = 2^{64} / 2^{30} GB = 2^{34} GB

b) ¿ Can the number 2^{63} be represented on this computer, reason for your answer?

In this computer, 64 bits are used to represent the numbers. With 64 bits the range of representation in two's complement is [2^{63}, 2^{63} -1], therefore it is not possible to represent that number.

c) ¿ What is the program counter and what is it used for? If a number is stored in the program counter in pure binary, what would be the range of representation?

The program counter is a processor register that stores the address of the next instruction to be executed. The range of representation would be $[0..2^{64}-1]$

Exercise 5. Indicate the correct decimal value corresponding to the following numbers represented in IEEE754 format in a briefly reasoned manner

a) 0xC1340000

1100 0001 0011 0100 0000 0000 0000 0000
0x 4 1 3 4 0 0 0 0

Sign = 1 (negative)
Exponent = 10000010 (130 – 127 = 3)
Mantissa= 01101 (with implicit biut)

$1{,}01101_2 \times 2^3 = 1011{,}01_2 \times 2^0 = -11{,}25_{10}$

Exercise 6. Given the following coce:

```
int vec[100];
int b = 5;

main ()
{
      int i = 0;
      for (i = 0; i < 100; i++)
            vec[i] = b+i;
}
```

write the MIPS32 assmbly language equivalent clearly, concisely and correctly

```
.data
      .aling 2 #next data aligned to 4
      vec: .space 400
      b:   .word 5

.text
            .glob main

main:
            li   $t0, 5
            la   $t1, vec
            li    $t2, 0
            li    $t3, 100
            li    $t4, 0
loop:       bge  $t2, $t3, end
            sw    $t0, vec($t4)
            addi $t2, $t2, 1
            addi $t4, $t4, 4
end:        b     loop
```

Exercise 7. Consider a 15-bit computer with a set of 12 machine instructions and a bank of 8 registers. Indicate an instruction format to be able to code the lw instruction of the form: `lw $t1 8($t2)`.

Solution:

4 bits	3 bits	3 bits	5 bits
Código de instrucción	Registro	Registro	relleno

Word 1

15 bits
desplazamiento

Woird 2

Exercise 8. A student wants to access the element matrix[i][j], an element of a 3 row by 5 column matrix. It is asked to answer correctly the piece of assembly code to make this access if the variable i that indicates the row is in $t1, and the variable j is in $t2.

solution:

```
# (i*5+j)*4
mul $t3 $t1 5
add $t3 $t2
mul $t3 $t3 4

# *( matriz + (i*5+j)*4 )
la  $t4 matriz
add $t4 $t4 $t3
lw  $t4 ($t4)
```

Exercise 9. Given the following function, indicate what parameters this subroutine has, what results it returns and what the routine does (you can use a mathematical expression depending on the input parameters or a description of the result obtained).

```
X:      li   $t5 0x8000
        sl   $t5 $t5 16
        and  $t2 $a0 $t5
        li   $t5 0x7F80
        sl   $t5 $t5 16
        and  $v0 $a0 $t5
        srl  $t2 $t2 31
        srl  $v0 $v0 23

        li   $t7 0x0007
        sl   $t7 $t7 16
        liu  $t0 0xFFFF
        or   $t7 $t7 $t0
        addi $v0 $v0 -127
        and  $v1 $a0 $t7
        beq  $t2 $0 fin1
fin1:   jr   $ra
```

Solution:

```
Function name: X

Parameters:
     $a0 -> numbder

Results:
     $v0 -> exponent without excess 127
     $v1 -> mantissa without sign
```

Exercise 10. Given the following funtioncs:

```
int F1 ( int a, int b, int c, int d, char e, char f ) {
       int ret[10] ;
       ret[2] = (a+b+c+d+(int)e+(int)f);
       return ret;
```

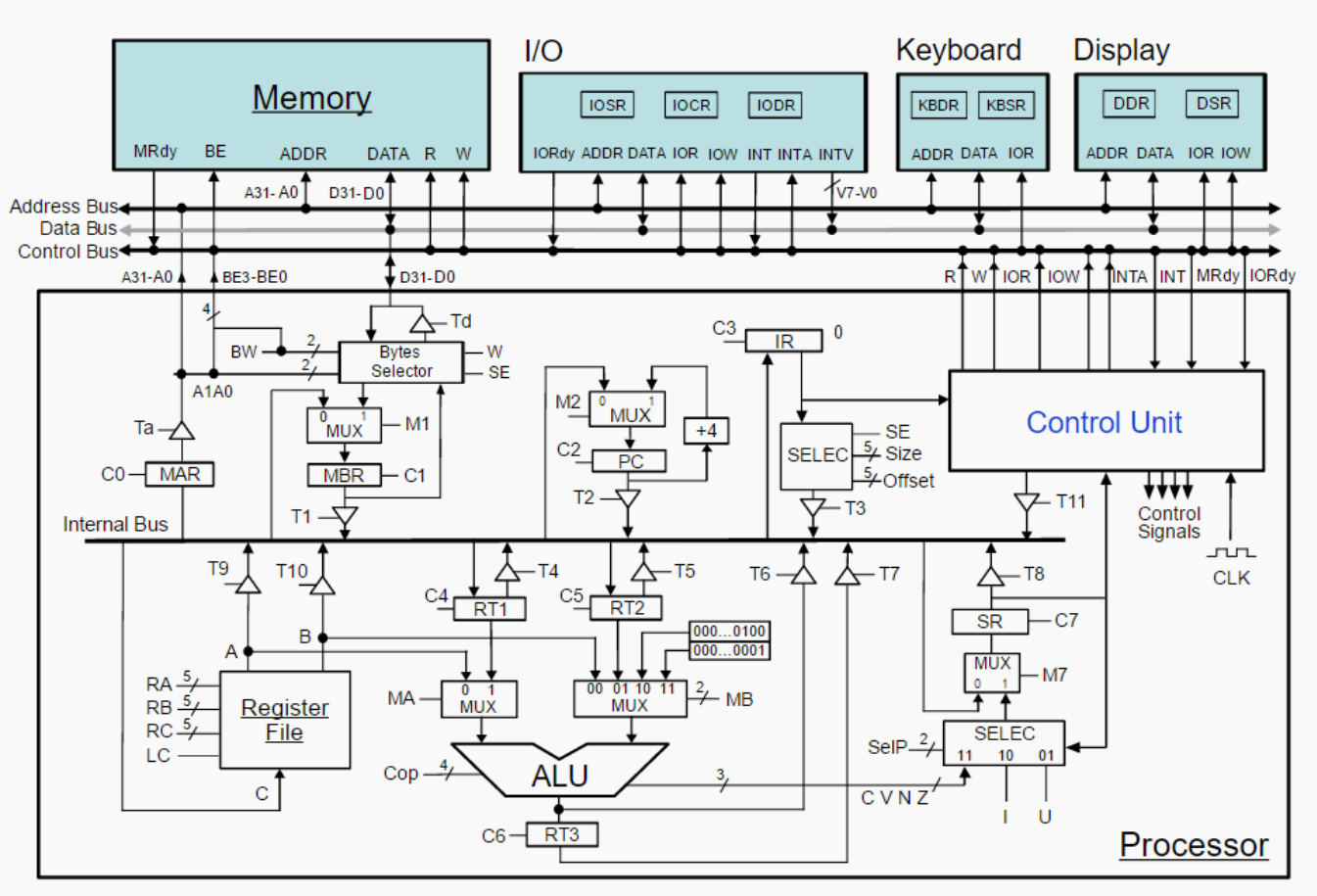
```
}

int main ( int argc, char *argv[] ) {
       F1(1,2,4,5,'a','b');
}
```

Write these routines in MIPS32 assembler following the parameter step agreement seen in class (and available in the published material). NOTE: it is not necessary to use the stack frame register.

Solución:

```
main:
     # push $ra
     subu $sp $sp 4
     sw $ra ($sp)

     # F1(1,2,4,5,'a','b')
     li $a0 1
     li $a1 2
     li $a2 4
     li $a3 5
     subu $sp $sp 8
     li $t0 'b'
     sw $t0 4($sp)
     li $t0 'a'
     sw $t0 0($sp)
     jal F1
     addu $sp $sp 8

     # pop $ra
     lw $ra ($sp)
     addu $sp $sp 4

     # return
     jr $ra


F1:
    # stack <- int ret
    subu $sp $sp 40

    add $t0 $a0 $a1
    add $t0 $v0 $a2
    add $t0 $v0 $a3

    lw $t1 4($sp)
    add $t0 $t0 $t1
    lw $t1 8($sp)
    add $t0 $t0 $t1

    # ret
    sw $t1 8($sp)
    move $v0 $sp

    # stack
    addu $sp $sp 40
    # return
    jr $ra
```

Exercise 11. Given the following processor:



Specify the elementary operations and control signals required to execute the machine instruction `push R1`. This instruction stores the value contained in the R1 register at the top of the stack, resetting the stack pointer register properly. Include the fetch cycle.
NOTE: Assume that R29 is the stack pointer (it points to the top of the stack and grows towards decreasing memory directions) and that memory is addressed by byte (and takes 1 cycle to operate).

Solution:

Elementary operations	Control signals
MAR <- PC	T2, C0
M[MAR] -> MBR, PC <- PC+4	Ta, R, BW=3, M1=1, C1
MBR -> RI	T1, C3
Decoding	Deco
Jump to Operation Code	Jump to Operation Code
MAR <- R29	RA=R29, T9, C0
MBR <- R1	RA=R1, T9, M1=0, C1
M[MAR] <- MBR	Ta, Td, BW=3, W
R29 <- R29 + 4	RA=29, MA=0, MB=4, Cop=+, T6, RC=R29, LC
Jumpt to fetch	Jumpt to fetch

Exercise 12. Write a routine in MIPS32 assemblñy called `count` to which you pass two parameters: the address of a square array of integers m, the number n of elements, and it returns the number of even numbers that are in the array. It has to use a double loop, as well as the convention of passing parameters and return seen in class.

Solution:

```
count:
        # result = 0
        li   $v0 0

        # for (i=0; i<$a1; i++)
        #       for (j=0; j<$a1; j++)
        li   $t1 0
   b1: bge  $t1 $a1 f1
        li   $t2 0
   b2: bge  $t2 $a1 f2

        # t3 = *($a0+(i*$a1+j)*4)
        mul  $t3 $t1 $a1
        add  $t3 $t3 $t2
        mul  $t3 $t3 4
        add  $t3 $t3 $a0
        lw   $t3 ($t3)

        # if ($t3 %2) == 0
        #     result++
        rem   $t3 $t3 2
        beqz  $t3 odd
        addi  $v0 $v0 1
odd:

        addi $t2 $t2 1
        b b2
   f2: addi $t1 $t1 1
        b b1
   f1: jr $ra                # return
```

Exercise 13. Briefly answer (in the space provided) the following questions:

a) What is the cache memory?

A lower capacity memory that offers more access speed (less latency).
It is placed between the CPU and the existing main memory so that only the part of the program that is being used at a given time is in cache memory.

b) What is an I/O module?

It is an element placed between the peripheral and the CPU, so that it provides an interface to the CPU that hides the peculiarities of the peripheral (different word sizes, timing, error control, etc.).

Exercise 14. Given a CPU accessing a cache of 10 ns access time in case of hit and an additional 100 ns transfer time in case of miss, what would be the average access time if the hit ratio is 90%?

tm = tca * h + (tca + tp) * (1 – h)
tm = 10 * 0.9 + (10 + 100) * (1 – 0.9) = 9 + 110 * 0.1 = 9 + 11 = 20 ns
tm = 20 ns

Exercise 15. Considere a device and the associated I/O module using a MIPS 32 architecture with separate input and output map and programmed input and output technique. The I/O module has three 32-bit registers:

- Data register (with address 0x108). It stores the position read.
- Control register (with address 0x104). When the value 2 is written in the register, the sensor is initialized.
- Status register (with address 0x100). If the register value is 0, no head position reading has been done. If the value is 1, a reading has been done and the controller has in the register the position data. If the value is -1, an error has occurred and you have to reset the reader (you only have to reset the reader once, unless an error occurs).

Develop a routine in assembly that will be in charge of initializing the reader and performing indefinitely the reading of codes. he value of the read code is printed on the screen (using the syscall service with code 1).

```
        # hardware init
ini:    li    $t0 2
        out $t0 0x104

        # read control record until it is non-zero
        bwait:  in     $a0 0x100
        beqz $a0 bwait

        # if there is an error, restart the hardware
        li          $t0 -1
        beq $a0 $t0 ini

        # print the value read and read again
        in $a0 0x108
        li $v0 1
        syscall

        b bwait
```