Name	

Matrikel-nummer	

Exam duration: 120 Minutes.
Tasks are to be solved directly on the exam sheets.
Minimum number of points to pass the exam: 40

Maximum score: 80 Points. ***Good Luck!***

Exercise 1 (10 points)

Answer the following statements (Correct answer: +1 point; No answer: 0 points; Wrong answer: -1 point).

a) In a word in 'Big-Endian' format, the left most byte is the most significant one.
Yes ☒ No ☐

b) The MIPS Register File contains sixty-four 32-bit registers.
Yes ☐ No ☒

c) In a Multi Cycle Architecture, all instructions take the same amount of time to execute.
Yes ☐ No ☒

d) The Program Counter (PC) is always incremented by 4 bytes.
Yes ☐ No ☒

e) In the case of 'loads' immediately followed by 'stores', stalling can be avoided by forwarding.
Yes ☒ No ☐

f) Hazard detection Unit controls the writing of the PC (PC.write) and IF/ID (IF/ID.write) registers.
Yes ☒ No ☐

g) 'Predict not taken' does not work well for 'top of the loop' branching structures.
Yes ☐ No ☒

h) A non-blocking cache allows the datapath to continue to access the cache while the cache is handling an earlier miss.
Yes ☒ No ☐

i) The maximum bus speed is largely limited by the length of the bus and the number of devices on the bus.
Yes ☒ No ☐

j) A synchronous bus involves more logic than an asynchronous bus.
Yes ☐ No ☒

Exercise 2 (6 Points)

a) (5 points) Suppose we have a processor with an ideal CPI of 1.5. The clock rate amounts to 1.6 GHz. A main memory access requires 30 ns. The proportion of load/store instructions amounts to 20% with a data cache failure rate of 4%. The failure rate for the instruction cache amounts to 3%.

   i. (3 points) What is the overall CPI?

Your answer:

CPI = 1.5 (cycle) + .2 (prop of load/store) x 0.04 (failure rate) x 30ns (memory delay) x 1.6 GHz (clock cycles) + 0.03 (failure rate to load any instruction) x 30ns (memory delay) x 1.6 GHz (clock cycles) = 1.5+0.384+1.44=3.32

   ii. (2 points) Suppose there is also a Level 2 cache added with an access time of 20 cycles for hit and miss. The error rate on the main memory is reduced to 0.05%. What is the overall CPI of this system (with an ideal CPI of 1.5)?

Your answer:

CPI_L1toL2 = .2 (prop of load/store) x 0.04 (failure rate) x 20 (cycles) + 0.03 (failure rate to load any instruction) x 20 (cycles) = 0.096

CPI_L2toMain = .2 (prop of load/store) x 0.0005 (failure rate) x 30ns (memory delay) x 1.6 GHz (clock cycles) + 0.0005 (failure rate to load any instruction) x 30ns (memory delay) x 1.6 GHz (clock cycles) = 0.0288

CPI = 1.5 (cycle) + CPI_L1toL2 + CPI_L2toMain = 1.5+0.096+0.0288 = 1.6248

Name

Exercise 3 (10 Points)

Suppose a MIPS processor with the simple 5-stages (IF, ID, EX, MEM and WB) pipeline is given. The instruction memory and data memory are separate and the register file can support one simultaneous read/write operation in one clock cycle. The processor supports forwarding only from ALU to ALU. In the case of the branch instruction, the next instruction to feed into the pipeline becomes known when the execute stage of the branch instruction has completed. Branch prediction is not supported. In the absence of hazards, a new instruction can be fed to the pipeline every cycle.

Consider the following MIPS Code

```
        1 add  $t1, $t2, $t3
        2 lw   $s2, 200($t1)
        3 add  $s3, $t1, $s2
        4 beq  $s3, $s3, L1
        5 sw   $s2, 200($t1)
L1:     6 addi $s1, $s1, 4
```

i. (10 points) How many cycles does this code take to complete? Show in a diagram the schedule in which the stages of the instructions in the above MIPS code are executed. Also indicate explicitly where forwarding or a simultaneous R/W operation can be used to resolve hazards.

Your answer:

	1	2	3	4	5	6	7	8	9	10	11	12	13
add	IF	ID	EX	M	RW								
lw		IF	ID	EX	M	RW							
add			IF	*	*	ID	EX	M	RW				
beq				IF	*	*	ID	EX	M	RW			
L1:addi									IF	ID	EX	M	RW

Exercise 4 (10 points)

a) (10 points) Given the following code fragment

```
do{
    g=g+A[i];
    i=i+j;
  } while (i != h)
```

where `A[]` is an array of integers, translate the code fragment to MIPS Assembly code. Assume that the variable `g` is in `$s1, h` is in `$s2, i` is in `$s3, j` is in `$s4 and 'base address of A'` is in `$s5.`

Your answer:

```
LOOP: sll  $t1, $S3, 2          # $t1=4*i
      add  $t1, $t1, $S5        # $t1=addrA
      lw   $t1, 0($t1)          # $t1=A[i]
      add  $S1, $S1, $t1        # $g=g+A[i]
      add  $S3, $S3, $S4        # i=i+j
      bne  $S3, $S2, LOOP       # go to LOOP if i!=h
```

Exercise 5 (10 points)

Answer the following questions:

a) (3 points) Name 3 types of buses and 2 main characteristics for each of them.

Your answer:

1. Processor – memory bus
   - Short and high speed
   - Optimized for cache block transfers
   - Matched to the memory system to maximize the memory processor bandwidth
2. I/O bus
   - Lengthy and slower
   - Needs to accommodate a wide range of I/O devices
   - Connects to the processor – memory bus or backplane bus
3. Backplane bus
   - The backplane is an interconnection structure within the chassis
   - Used as an intermediary bus connecting I/O busses to the processor – memory bus
   - It is designed to allow processors, memory, and I/O devices to coexist on a single bus so it has the cost advantage of having only one single bus for all components

b) (2 points) We have a program consisting of a conditional branch. The outcomes of the branch for an execution of the program are as follows: T-N-T-T-T-N-T-N-T (T stands for Taken and N for Not Taken). Calculate the predictions and accuracies for the following two branch prediction schemes.

i. (1 point) 1-bit predictor, initialized to predict taken.

Your answer:

T-T-N-T-T-T-N-T-N

ii. (1 point) 2-bit predictor, initialized to predict taken.

Your answer:

T-T-T-T-T-T-T-T-T

c) (2 points) What are the definitions for latency and bandwidth?

Your answer:

Latency: time to access one word.
Bandwidth: how much data from the memory can be supplied to the processor per unit time.

d) (1 point) In MIPS unconditional branch instructions, how is the destination address specified?
(PROVIDE THE BIT FORMAT)

Your answer:

Syntax:	j target
Encoding:	0000 10ii iiii iiii iiii iiii iiii iiii

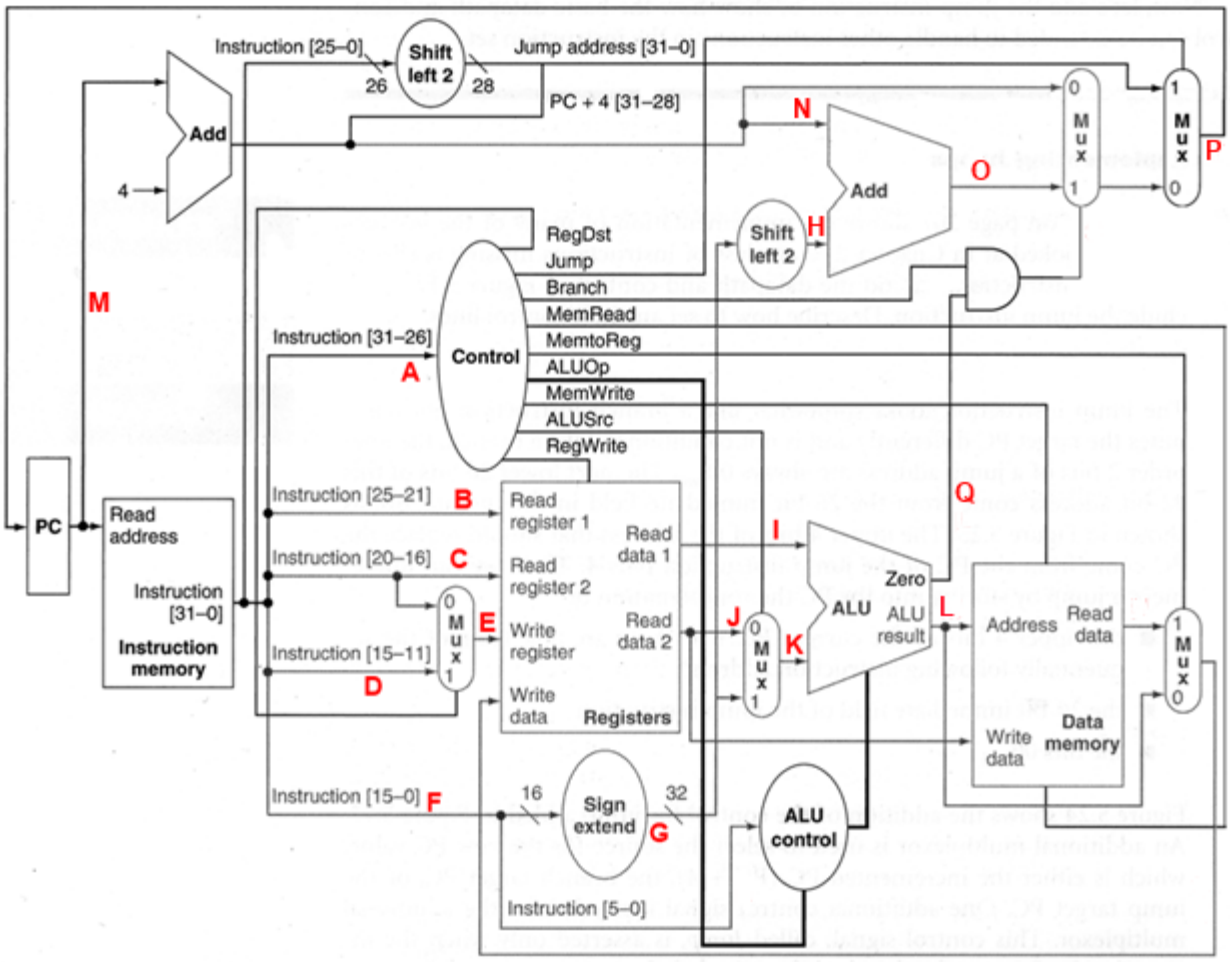
e) (2 points) What is the Branch Target Buffer used for?

Your answer: Control hazards, page 20

Exercise 6 (15 points)

You are given the following single cycle implementation of the MIPS ISA. Assume that a program (at instruction memory address `0x0000'2024`) executes the instruction `beq r20,r23,-0x7`. The state of the register file is given in the following table:

Register file	
Register	**Content**
`PC`	`0x0000'2024`
`$r20`	`0x0000'001C`
`$r21`	`0x0000'1008`
`$r22`	`0x1004'006A`
`$r23`	`0x0000'001C`



Note: For this exercise the `0x` prefix is used to indicate hexadecimal values.

Based on the given implementation and register states, answer the following questions:

a) (5 points) How is the instruction `beq r20,r23,-0x7` encoded? State the binary representation of this instruction and document your solution process. The opcode of `beq` is `0x04`.
(Hint: `beq` is an I-type instruction, the syntax of an `beq` instruction is `beq $rs,$rt,offset`)

Your answer:

```
opcode = 4 = 0x04 = 0b0000'0100
rs = 20 = 0x14 = 0b0001'0100
rt = 23 = 0x17 = 0b0001'0111
immediate = offset = -7 = 0xFFF9 = 0b1111'1111'1111'1001
(7 = 0b0000'0000'0000'0111, 7 = 0b1111'1111'1111'1001)

I-Type:    opcode(6)|rs(5)|rt(5)|immediate(16)
Result:    000100|10100|10111|1111'1111'1111'1001
           0x04|0x14|0x17|0xFFF9
           4|20|23|-7/65529(unsigned)
```

b) (8 points) For each letter in the diagram (A…Q), state the value of the signal on the corresponding wire / bus. You may use binary, hexadecimal or decimal values. All (if any) undefined signals are to be marked with `X`.
Additionally, what are the values of the control signals `RegWrite, Branch` and `Jump`?

Your answer:

A	**B**	**C**	**D**	**E**
0x04 (op)	0x14 (rs)	0x17 (rt)	0b1'1111	?
F	**G**	**H**	**I**	**J**
0xFFF9 (imm)	0xFFFF'FFF9 (s.e. imm)	0xFFFF'FFE4 (G << 2)	0x1C r[20]	0x1C r[23]
K	**L**	**M**	**N**	**O**
->J	0 (0x1C-0x1C)	0x2024 (pc)	0x2028 (pc+4)	0x200C (pc+4+(-7)*4)
P	**Q**	**RegWrite**	**Branch**	**Jump**
->O	1 (0x1C==0x1C)	0	1	0

Exercise 7 (20 points)

a) (2 points) Explain the meaning of the following variable declarations:

i. `int *ptr[30];`
ii. `int *a, b;`
iii. `float const * const f;`
iv. `void*(*foo)(int*);`

Solution:

i) ptr is an Array of 30 integer pointer.
ii) a is a pointer to integer, b is an integer
iii) f is a constant pointer to a constant float value.
iv) foo is a function pointer to a function which takes a pointer to int as argument and returns a void pointer.

b) (1 point) What does the `typedef` keyword do? Explain and give an example usage (no code required).

Solution:

The "typedef" keyword can be used to define a new name for an existing variable type. A typical use is to typedef a struct to avoid always typing "struct myStruct a". Another use is typedef a function pointer to increase readability.

c) (1 point) What is the use of a void pointer as function argument?

Solution:

Using a void pointer allows to pass pointers to different kinds of data to a function. The pointer can then be cast to the actual type before using.

d) (4 points) What is the output of the following program?

```
#include<stdio.h>

#define CUBE(x) (x*x*x)

int main(void) {
    int b = 3;
    int a = CUBE(b++);
    printf("%d, %d\n", a, b);
    return 0;
}
```

Solution
60, 6

e) (4 points) What output do you expect from the following program? Does the algorithm build a valid copy of the vector `a`? If not, how can the error be fixed?

```
#include <stdio.h>

int* create_copy(int *a, int size) {
    int new[size];
    for (int i = 0; i < size; ++i) {
        new[i] = a[i];
    }
    return new;
}

#define ARRAY_SIZE 10
int main(void) {
    int a[ARRAY_SIZE] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

    int *b = create_copy(a, ARRAY_SIZE);

    a[0] = 12;

    for (int i = 0; i < ARRAY_SIZE; ++i) {
       printf("%d ", a[i]);
    }
    printf("\n");
    for (int i = 0; i < ARRAY_SIZE; ++i) {
        printf("%d ", b[i]);
    }
    return 0;
}
```

Solution:

Expected output:
12 2 3 4 5 6 7 8 9 10
some garbage values

The output is incorrect, the second array contains garbage because it has been allocated on the stack inside the function. Therefore the pointer is invalid after the execution returns from the function. The inteded output would be:
12 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10

The code can be fixed by replacing int new[size] with:
int* new = malloc(size * sizeof(int));

f) (8 points) Assume that the variable `m` in the following code fragment points to a square matrix stored in row major order and `dim` stores its dimension.

Example: the matrix $\begin{pmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{pmatrix}$ is stored as linear array 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 in memory and dim == 3.

What does the following code do? You can give a high level description or just simulate the function on the given example matrix from above and describe the output.

```
void func(int* m, int dim) {
   for (int r = 0; r < dim; ++r) {
      for (int c = r + 1; c < dim; ++c) {
         int *ptr1 = m + r * dim + c, *ptr2 = m + c * dim + r, tmp = *ptr1;
         *ptr1 = *ptr2;
         *ptr2 = tmp;
      }
   }
}
```

Solution:

The function transposes the given matrix, e.g.:

$$\begin{pmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{pmatrix} \rightarrow \begin{pmatrix}1 & 4 & 7\\2 & 5 & 8\\3 & 6 & 9\end{pmatrix}$$