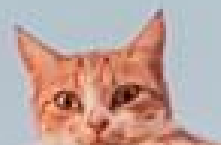

Computer Orgnization

CS202-2023s　　CPU 大作业要求
CPU major task requirements

总体说明

开发板领用说明：**每小组一块开发板，请保护好开发板，如有丢失或损坏需照价赔偿。**
开发板借用之后请尽快完成基本测试 **(建议一周内完成以便及时发现问题)**，如有问题请反馈具体问题并及时找老师更换。

组队规则：
必须保证在**答辩时间内全员到场。**
建议同一个实验班的同学组队，也可以同一个理论班下不同实验班的同学跨班组队。
3人组队（如果人数不够也可以2人组队，但不建议）。

团队得分 = 功能验收分 * **系数** + 代码规范分 + 文档得分 + bonus得分
个人得分 = 团队得分 * 团队人数 * 个人贡献百分比

答辩 说明	代码提交	文档（视频）提交	系数
提前答辩（15周实验课时间）	答辩前	16周周一前	1.05
正常答辩（16周实验课时间）	答辩前	17周周一前	1
推迟答辩（16周周五，17周）	答辩前	答辩的后一周周一前	0.6~0.9

系数 说明：如代码、文档（视频）任何一个提交件延迟提交，**系数** 将按最晚的提交时间为准。 比如：A小组在15周完成答辩，如所有交付件按照以上表格的要求准时提交，则**系数为1.05；如**代码准时提交但开发文档在16周周二提交，**系数** 按最晚提交的开发文档的时间来计算，对应系数**为1。**

提交要求

- 提交要求（每小组只需要提交一份，分两次提交，两次提交都应是同一位组员）：
  - **第一次提交源代码（答辩前提交bb站点）**
    - 代码：含CPU设计文件（包括IP核说明文件xci）、仿真用的testbench文件（可选，仅不能完成上板测试的同学才必须做提交）、上板测试用的约束文件、测试场景对应的asm以及coe文件（用于查重）
    - 压缩包的名字格式为：　　**c答辩时间_小组成员姓名列表**

    比如： c16178_A_B_C (其中c16178表示16周周一78节课上答辩做的代码提交，A,B,C是三名队友的名字)

  - **第二次提交文档及视频（答辩当周的下周周一之前）**
    - 文档（pdf格式），文档名：**d答辩时间_小组成员姓名列表，如无视频则只提交文档即可**
    - 视频（可选）：
      - 争取bonus的小组须针对bonus部分录制项目功能演示视频，不争取bonus的小组对视频不做要求
      - 请将文档和视频放一个文件夹压缩后提交，压缩包的名字格式为：**dv答辩时间_小组成员姓名列表**

评分说明（1）

- 评分以代码规范（结构化设计、变量命名、代码规范、注释）、文档、功能验收演示为准。
- 项目得分包括两个部分：基本分+ bonus，如得分超过100，则溢出的部分将按比例计入总评 .
  - 基本分：基本功能 + 代码规范 +文档
    - 基本功能：
      - 1）使用外设的种类数>2
        （按键开关和拨码开关属于两种不同类型输入设备，led和七段数码显示管属于两种不同类型的输出设备）
      - 2）测试通过基本场景1、基本场景2
  - bonus：功能 +文档 +视频
    - bonus中合格的文档、视频的分数都按照以下方式计算： bonus功能分*0.25
    - **请注意：如缺少bonus对应的文档，除对应的文档分为0外，bonus的功能分要打八折。**
  - 补充说明：如果不能上板测试，则根据情况，项目总分*(0.3~0.6)

评分说明（2）

- bonus 包括但不限于：
  - 1）实现对复杂外设接口的支持（如VGA接口、小键盘接口等）
    - 说明：在本课程中，仅支持**通过软硬件协同的方式实现的复杂外设接口的访问（**即或者通过相应的指令，或者通过指令中相应的地址信息来访问相关的复杂外设，而不仅仅是以硬件控制的方式来实现对复杂外设的使用）。
  - 2）使用uart接口实现不重新烧写FPGA芯片的前提下加载不同的程序到CPU上做执行。
  - 3）基于现有 Minisys ISA 实现的CPU，实现性能的优化（pipeline、cache、SIMD等）
    - 说明：需同时提供性能提升的相关比对用例及证明。
  - 4）针对现有Minisys 的ISA，指令类型扩展、功能扩展和实现。
  - 5）更好的用户体验
    - 比如 位运算的结果用led显示，算术类运算结果用7段数码管显示 等

- 补充1：实现其他类型的ISA（如RISC-V，MIPS32）将按照以上要求进行检查，根据完成情况在bonus的总分基础上乘以1.05~1.2的系数（该情况下，不重复考虑bonus的3、4部分）。**请注意**，相关**文档中必须比对**该体系结构的实现与课上介绍的Minisys的实现之间的**差异，否则相关bonus为0分。**
- 补充2：所有实现的bonus部分，必须在**文档**中补充关于bonus的实现机制、测试用例以及测试结果的说明，并录制相应的视频对bonus对应的内容进行演示。如文档中没有相关说明，则对应的文档分数为0，且bonus对应的功能分*0.8。

文档要求（1-基本分文档）

- 开发者说明：每个成员的学号、姓名、所负责的工作、贡献百分比。
- 版本修改记录（可选）：版本号、时间、更新点描述
- CPU架构设计说明
  - CPU特性：
    - ISA（含所有指令（指令名、对应编码、使用方式），参考的ISA，基于参考ISA本次作业所做的更新或优化；寄存器（位宽和数目）等信息）；对于异常处理的支持情况。
    - 寻址空间设计：属于冯.诺依曼结构还是哈佛结构；寻址单位，指令空间、数据空间的大小。
    - 对外设IO的支持：采用单独的访问外设的指令（以及相应的指令）还是MMIO（以及相关外设对应的地址），采用轮询还是中断的方式访问IO。
    - CPU的CPI，属于单周期还是多周期CPU，是否支持pipeline（如支持，是几级流水，采用什么方式解决的流水线冲突问题）。
  - CPU接口：时钟、复位、uart接口（可选）、其他常用IO接口使用说明。
  - CPU内部结构
    - CPU内部各子模块的接口连接关系图
    - CPU内部子模块的设计说明（模块功能、子模块端口规格及功能说明）
- 测试说明：以表格的方式罗列出测试方法（仿真、上板）、测试类型（单元、集成）、测试用例描述、测试结果（通过、不通过）；以及最终的测试结论。
- 问题及总结：开发过程中遇到的问题、思考、总结。

和bonus相关的文档及视频要求

- 和bonus相关的文档要求
  - 和bonus相关的说明请放在基本功能文档的后半部分。
  - bonus 对应功能点的设计说明
    - 设计思路及与周边模块的关系
    - 核心代码及必要说明
  - 测试说明：测试场景说明，测试用例，测试结果及说明。
  - 问题及总结：在bonus功能点开发过程中遇到的问题、思考、总结。

- 和bonus相关的视频要求:
  - 视频中需要有本次大作业的完整介绍（包括小组成员，整体功能，尤其是bonus相关功能点）
  - 主体内容为：bonus的设计思路介绍、功能演示及说明

答辩要求

- 答辩前准备：
  - 设备：请准备两台安装有vivado的电脑参与答辩（需现场修改汇编代码，烧写fpga芯片，对照代码回答问题，两台电脑方便同步开展测试）。
  - 答辩次序登记：在共享文档中登记答辩时间、答辩次序。
- 答辩包括：
  - 演示、问答两个环节，所有组员都必须到场并回答问题。
  - 要求现场根据演示要求修改汇编源代码，完成汇编、下发程序、测试的完整过程。
- 演示过程中需按要求完成CPU的上板（Minisys/EGO1开发板）测试。
  - CPU的基本测试场景（参见后页具体内容）
  - CPU的扩展功能（参考p5）

基本测试场景1

说明：使用开发板上的3+8个拨码开关用于做输入，其中3个拨码开关(x2,..x0) 用于测试用例的编号输入，8个拨码开关（sw7,..sw0) 用于做测试数据的输入，使用led灯或者7段数码显示管做输出.

场景1.用例编号	用例描述
3‘b000	**输入**测试数**a**，输入完毕后在led灯上**显示a**，同时用1个led灯显示a是否为**二的幂**的判断 （比如8’h01，8‘h10是二的幂，该led灯亮，8’ha0,8’h0a不是二的幂则该led灯不亮)
3‘b001	**输入**测试数**a**，输入完毕后在输出设备上**显示a**，同时用1个led灯显示a是否为**奇数**的判断 （比如8’h01，8‘hab是奇数，该led灯亮，8’ha0,8’hbc不是奇数则该led灯不亮)
3‘b010	**先执行测试用例3’b111**, 再计算 **a 和 b的按位或**运算，将结果显示在输出设备
3‘b011	**先执行测试用例3’b111**, 再计算 **a 和 b的按位或非**运算，将结果显示在输出设备
3‘b100	**先执行测试用例3’b111**, 再计算 **a 和 b的按位异或**运算，将结果显示在输出设备
3‘b101	**先执行测试用例3’b111**, 再执行 **slt** 指令，将**a和b按照有符号数进行比较**，用输出设备展示a<b的关系是否成立 (关系成立，亮灯，关系不成立，灭灯)
3‘b110	**先执行测试用例3’b111**, 再执行 **sltu** 指令，将**a和b按照无符号数进行比较**，用输出设备展示a<b的关系是否成立 (关系成立，亮灯，关系不成立，灭灯)
3‘b111	**输入**测试数**a**，**输入**测试数**b**，**在输出设备上展示a和b的值**

基本测试场景2

- 使用开发板上的3+8个拨码开关用于做输入，其中3个拨码开关（x3-x0）用于测试用例的编号输入，8个拨码开关（sw7,..sw0）用于做测试数据的输入（sw7对应于8bit的最高bit位bit7，sw0对应于8bit的最低bit位bit0）；

场景2.用例编号	用例描述
3'b000	输入**a**的数值（a被看作**有符号数**），计算**1到a的累加和**，在输出设备上**显示累加和（如果a是负数，以闪烁的方式给与提示）**
3'b001	输入**a**的数值（a被看作**无符号数**），以**递归的方式计算1到a的累加和**，记录本次入栈和出栈次数，在输出设备上**显示入栈和出栈的次数之和**
3'b010	输入**a**的数值（a被看作**无符号数**），以递归的方式计算1到a的累加和，记录入栈和出栈的数据，在输出设备上**显示入栈的参数，每一个入栈的参数显示停留2-3秒 （说明，此处的输出不关注$ra的入栈和出栈信息）**
3'b011	输入**a**的数值（a被看作**无符号数**），以递归的方式计算1到a的累加和，记录入栈和出栈的数据，在输出设备上**显示出栈的参数，每一个出栈的参数显示停留2-3秒（说明，此处的输出不关注$ra的入栈和出栈信息）**
3'b100	输入测试数**a**和测试数**b**，实现**有符号数（a，b以及相加和都是8bit，其中的最高bit被视作符号位，如果符号位为1，表示的是该负数的补码**）的**加法**，并对是否溢出进行判断，**输出运算结果以及溢出判断**
3'b101	输入测试数**a**和测试数**b**，实现**有符号数（a，b以及差值都是8bit，其中的最高bit被视作符号位，如果符号位为1，表示的是该负数的补码**）的**减法**，并对是否溢出进行判断，**输出运算结果以及溢出判断**
3'b110	输入测试数**a**和测试数**b**，实现**有符号数（a，b都是8bit，乘积是16bit，其中的最高bit被视作符号位，如果符号位为1，表示的是该负数的补码**）的**乘法**，**输出乘积**
3'b111	输入测试数**a**和测试数**b**，实现**有符号数（a，b，商和余数都是8bit，其中的最高bit被视作符号位，如果符号位为1，表示的是该负数的补码**）的**除法**，**输出商和余数（商和余数交替显示，各持续5秒）**

Overall Description

Development board borrowing policies：Each group gets only one board. Please take care of the board otherwise you need to reimburse for any loss or damage based on its cost.
The development board is borrowed and the basic testing is completed within one week. If there are any issues, please provide feedback on the specific issues and promptly contact the teacher for replacement.

Team rules:
It is necessary to ensure that all members are present during the defense time. It is recommended that students from the same lab session form teams, or that students from different lab within the same lecture class collaborate across sessions. Ideally, teams should consist of 3 members (teams of 2 are allowed if there are insufficient participants but not recommended).

Team Score = Functionality Approval Score* **Coefficient** + Code Quality Score + Documentation Score + Bonus Score
Individual Score = Team Score * Team Size * Individual Contribution Percentage

demonstration	code submission	文档（视频）提交	Coefficient
advance（lab class in week15）	before demonstration	before Monday of week 16	1.05
nomal（lab class in week16）	before demonstration	before Monday of week 17	1
delay（lab class on Friday of week16, week 17）	before demonstration	before Monday of the week after the domonstration	0.6~0.9

Coefficient description: If any submission of code or documents (videos) is delayed, the coefficient will be determined by the using the latest submission time..
For example, if Group A completes the demonstration within 15 weeks and all deliverables are submitted on time according to the requirements of the above table, the coefficient is 1.05; If the code is submitted on time but the development document is submitted on Tuesday, the coefficient is calculated based on the latest submission time of the development document, and the corresponding coefficient is 1.

The requirements about submission

Each group only needs to submit one copy, divided into two submissions, with both submissions coming from the same team member:

- First submission - Source code (submit to the Blackboard site before the demonstration)
  - Code: Includes CPU design files (with IP core specification file xci), optional simulation testbench files (required only for students who can't complete board testing), board testing constraint files, and asm and coe files for testing scenarios (for **plagiarism checking**)
  - Compressed package name format: c_demo_time_name_list Example: c16178_A_B_C (c16178 represents the code submission for the 78th class demonstration on Monday of week 16, with A, B, and C as the three teammates' names)

- Second submission - Documents and videos (submit before the next Monday after the demonstration week)
  - Document (PDF format): Name format: d_demo_time_name_list. If there's no video, just submit the document.
  - Video (optional):
    - Teams aiming for a bonus must record project functionality demo videos for the bonus section. No video requirements for teams not seeking a bonus.
    - Compress the documents and videos into a folder and submit them. The compressed package name format is: dv_defense_time_name_list.

Scoring Instructions (1)

- Scoring is based on code specifications (structured design, variable naming, code specifications, annotations), documentation, and functional acceptance demonstrations.
- The project score consists of two parts: basic score and bonus . If the score exceeds 100, the overflow will be proportionally included in the overall evaluation
  - Basic score: Basic functions +Code specifications +Documents
    - Basic functions :
      - 1) Number of types of peripherals used>2

        (The key switch and Dip switch belong to two different types of input devices, and the led and seven segment digital display tube belong to two different types of output devices)
      - 2) Test passed Basic Scenario 1, Basic Scenario 2
  - bonus: Features +Documents +Videos
    - The scores for qualified documents and videos in bonus are calculated as follows: bonus function score * 0.25
    - **Please note that if a document corresponding to bonus is missing, in addition to the corresponding document being divided into 0, the functional score of bonus will be discounted by 20%.**
  - Supplementary note: If board testing is not possible, the total score of the project will be * (0.3~0.6) depending on the situation

Scoring Instructions (2)

- Bonus includes but is not limited to:
  - 1) Implement support for complex peripheral interfaces (such as VGA interfaces, mini keyboard interfaces, etc.)
    - Explanation: In this course, only access to complex peripheral interfaces achieved through software and hardware collaboration is supported (that is, accessing related complex peripherals through corresponding instructions or corresponding address information in instructions, rather than just using hardware control to achieve the use of complex peripherals)。
  - 2) Use the uart interface to load different programs onto the CPU for execution without rewriting the FPGA chip。
  - 3) Optimize performance based on existing Minisys ISA implemented CPUs (pipeline, cache, SIMD, etc.)
    - Explanation: Relevant comparison cases and proof of performance improvement need to be provided simultaneously。
  - 4) ISA for existing Minisys, instruction type extension, function extension, and implementation。
  - 5) Better user experience
    - For example, the results of bit operations are displayed with LED, and the results of arithmetic operations are displayed with 7 segments of Nixie tube, etc

- Supplement 1: Implementing other types of ISA (such as RISC-V and MIPS32) will be checked according to the above requirements, and based on the completion status, the total score of the bonus will be multiplied by a coefficient of 1.05-1.2 (in this case, parts 3 and 4 of the bonus will not be considered repeatedly). Please note that the differences between the implementation of this architecture and the Minisys implementation introduced in class must be compared in the relevant documents, otherwise the relevant bonus will be 0 points。
- Supplement 2: For all implemented bonus parts, explanations on the implementation mechanism, test cases, and test results of bonus must be included in the document, and corresponding videos must be recorded to demonstrate the corresponding content of bonus. If there is no relevant explanation in the document, the corresponding document score is 0, and the function score corresponding to bonus is * 0.8。

The requirements about document (1-basic part)

- Developer Description: Each member's student ID, name, job responsibilities, and contribution percentage。
- Version modification record (optional): version number, time, and update point description
- CPU Architecture Design Description
  - CPU features:
    - ISA (including all instructions (instruction names, corresponding encodings, usage methods), referenced ISA, and updates or optimizations made by referencing ISA in this assignment); Registers (bit width and number) and other information; Support for exception handling.
    - Address space design: belonging to von Neumann structure or Harvard architecture; Addressing unit, size of instruction space and data space。
    - Support for external IO: Use separate instructions to access the peripheral (and corresponding instructions) or MMIO (and corresponding addresses of the peripheral), and use polling or interrupt methods to access IO。
    - The CPI of the CPU, whether it is a single cycle or multi cycle CPU, does it support pipelines (if so, what level of pipeline is it, and what method is used to solve pipeline conflicts)。
  - CPU interface: clock, reset, UART interface (optional), instructions for using other commonly used IO interfaces。
  - CPU internal structure
    - Interface connection diagram of sub modules within the CPU
    - Design Description of CPU Internal Submodules (Module Functions, Submodule Port Specifications, and Functional Description)
- Test description: list the test method (simulation, board loading), test type (unit, integration), test case description, test results (pass, fail) in a table; And the final test conclusion。
- Problems and Summary: Problems encountered during the development process, reflections, and summaries。

Documentation and video requirements related to Bonus

- Documentation requirements related to bonuses
  - Please place the explanations related to bonus in the latter half of the basic function document。
  - Design instructions for the corresponding function points of bonus
    - Design ideas and their relationship with surrounding modules
    - Core code and necessary instructions
  - Test description: Test scenario description, test cases, test results and explanations。
  - Problem and summary: Problems encountered during the development process of the bonus function point, reflections, and summaries。

- Video requirements related to bonus:
  - The video requires a complete introduction to this major assignment (including team members, overall functions, especially the bonus related function points)
  - The main content is: Introduction to the design concept, functional demonstration, and explanation of bonus.

The requirements about demonstration

- Preparation for the demonstration：
  - Equipment: Please prepare two computers with Vivado installed to participate in the defense (assembly code needs to be modified on-site, FPGA chips need to be burned, and questions need to be answered by comparing the code, so that the two computers can conveniently conduct testing simultaneously)。
  - Registration of demonstration order: Register the defense time and defense order in the shared document。
- Demonstration：
  - demonstration、Q&A two parts, All team members must be present and answer questions。
  - Require on-site modification of assembly source code according to demonstration requirements, complete the complete process of assembly, program distribution, and testing。
- During the demonstration process, it is necessary to complete the board testing of the CPU (Minisys/EGO1 development board) as required。
  - Basic testing scenarios for CPUs (see specific content on the following page)
  - Expansion function of CPU (refer to p5)

Basic Test Scenario 1

Note: 3+8 switch on the development board are used for input, of which 3 switches (x2,.. x0) are used for number input of test cases, 8 switches (sw7,.. sw0) are used for input of test data, and LED lights or seven-segments display tubes are used for output

Scenario 1. Testase ID	Testcase Description
3‘b000	Enter the test number **a**, display **a** on the LED light. At the same time, use one LED light to determine **whether a is a power of two** (e.g. 8'h01 and 8'h10 are powers of two, the LED light is on. 8'ha0 and 8'h0a are not powers of two, the LED light is not on)
3‘b001	Input the test number **a**, display **a** on the output device. At the same time, use one LED light to display **whether a is an odd number**(e.g, 8'h01 and 8'hab are odd numbers, the LED light will be on. 8'ha0 and 8'hbc are not odd numbers, the LED light is not on)
3‘b010	Execute testcase 3'b111 first, then calculate the bitwise **OR** operation of **a** and **b**, and display the results on the output device
3‘b011	Execute testcase 3'b111 first, then calculate the bitwise **NOR** operation of **a** and **b**, and display the results on the output device
3‘b100	Execute test case 3'b111 first, then calculate the bitwise **XOR** operation of **a** and **b**, and display the results on the output device
3‘b101	First execute test case 3'b111, then execute the **SLT** instruction, **compare a and b as signed numbers**, and use the output device to demonstrate whether the relationship between a and b is valid.(Relationship established, light on, relationship not established, light off)
3‘b110	First execute test case 3'b111, then execute the **SLTU** instruction, **compare a and b as unsigned numbers**, and use the output device to demonstrate whether the relationship between a and b is valid(Relationship established, light on, relationship not established, light off)
3‘b111	Input test number a, input test number b, and display the values of a and b on the output device

Basic Test Scenario 2-1

3+8 switch on the development board are used for input, of which 3 switch (x3-x0) are used for the number input of test cases, and 8 switch (sw7,.. Sw0) are used for the input of test data (sw7 corresponds to the highest bit bit of 8bit bit7, sw0 corresponds to the lowest bit bit bit of 8bit bit0);

Scenario 2. Testase ID	Testcase Description
3‘b000	Enter the numerical value of **a** (**a is considered a signed number**), **calculate the cumulative sum of 1 to a,** and display the cumulative sum on the output device **(if a is a negative number, give a blinking prompt**)
3‘b001	Enter the numerical value of **a (a is considered an unsigned number), recursively calculate the sum of 1 to a**, record the number of times the stack was pushed and pushed, and **display the sum of the times the stack was pushed and popped on the output device**
3‘b010	Enter the numerical value of **a (a is considered an unsigned number)**, **recursively calculate the sum of 1 to a,** record the data of stack entry and exit, and **display the parameters which is pushed to the stack on the output device**. **Each parameter of the stack is displayed for 2-3 seconds** (indicating that the output here does not pay attention to the stack entry and exit information of $ra)
3‘b011	Enter the numerical value of **a (a is considered an unsigned number)**, **recursively calculate the sum of 1 to a,** record the data of stack entry and exit, and **display the parameters which is popped from the stack on the output device**. **Each parameter of the stack is displayed for 2-3 seconds** (indicating that the output here does not pay attention to the stack entry and exit information of $ra)

Basic Test Scenario 2-2

3+8 switch on the development board are used for input, of which 3 switch (x3-x0) are used for the number input of test cases, and 8 switch (sw7,.. Sw0) are used for the input of test data (sw7 corresponds to the highest bit bit of 8bit bit7, sw0 corresponds to the lowest bit bit bit of 8bit bit0);

Scenario 2. Testase ID	Testcase Description
3‘b100	Input test number **a** and test number **b** to implement the **addition of signed numbers** (**a, b, and the sum of additions are all 8 bits, where the highest bit is considered the sign bit. If the sign bit is 1, it represents the 2’s complement of the negative number**), and determine whether overflow occurs. **Output the operation result and overflow judgment**
3‘b101	Input test number **a** and test number **b** to **subtract signed numbers** (**a, b, and the difference are all 8 bits, where the highest bit is considered as the sign bit. If the sign bit is 1, it represents the 2’s complement of the negative number**), and determine whether overflow occurs. Output the operation result and overflow judgment
3‘b110	Input test number **a** and test number **b** to implement **the multiplication of signed numbers (a and b are both 8 bits, the product is 16 bits, and the highest bit is considered as the sign bit. If the sign bit is 1, it represents the 2’s complement of the negative number)**, and **output the product**
3‘b111	Input test number **a** and test number **b** to achieve **division of signed numbers (a, b, quotient and remainder are both 8 bits, where the highest bit is considered the sign bit. If the sign bit is 1, it represents the complement of the negative number**), and **output quotient and remainder (quotient and remainder are displayed alternately, each lasting for 5 seconds)**