Computer Orgnization

CS202-2024s CPU project

Overall NOTES（IMPORTANT！）

Instructions for **development boards**: Each group has one development board. Please keeping the development board and compensate according to the price if it is lost or damaged.
The basic testing of the development board should be completed within one week of borrowing. If there are any problems, please provide specific feedback and promptly contact the teacher for replacement.。

Teaming rules: 1）It is necessary to ensure that all members are present during the defense time. If not present, the number of team members will be automatically reduced, and the contribution ratio will be arranged according to the actual number of people.
2）It is recommended that students from the same experimental class form teams, or that students from different experimental classes in the same theoretical class form teams across classes.
3）Team up with 3 people (if the number is not enough, you can also team up with 2 people, but it is not recommended).
4）The minimum and maximum contribution ratios shall not exceed 10%, that is, the maximum ratio for a two person group is 45:55, and the maximum ratio for a three person group is 30:30:40.

Personal Project Overall Evaluation=Team Score * Team Size * Individual Contribution Percentage+Individual Defense Performance Score (15)
Team score=Functional score (70) * Delay/Advance coefficient (0.6~1.05)+Code specification score (3)+Document score (12)+Bonus score (15)

defense time	submit code	submit document(vedio)	advance/delay Coefficient （multiply with function score）
advanced （experimental class in week15）	before 12:00 am on Monday of week15	before 12:00 am on Monday of week16	1.05
nomal （experimental class in week16）	before 12:00 am on Monday of week16	before 12:00 am on Monday of week17	1
delay（arranged uniformly after all delayed groups finish submission）	One day late with a 5% deduction factor	before 12:00 am on Monday of week17	0.9 for Tuesday on week16，0.85 for Wednesday on week16, 0.8 for Thursday on week 16，0.75 for Friday on week 16，0.7 for saturday on week16, 0.65 for Sunday on week16，0.6 for Monday on week17，code must be submit before 12:00 am on the day.

Coefficient explanation: If any submission of code, document (video) is delayed, the coefficient will be calculated based on the latest submission time to calculate the "functional acceptance score". For example, if Group A completes the defense 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.
Personal overall evaluation explanation: If the contribution ratio of the three person group is 1:1:1 and the defense takes 15 weeks, the highest score can be obtained (70 * 1.05+3+12+15) * 3 * 0.33+15=118.5

Submission requirements

- submission requirement (Each group only needs to submit one copy, divided into two submissions, both of which should be from the same team member) :
  - **1st submission: source file** (Submit to the BlabckBoard site before 12:00 AM on Monday of the week of defense)
    - source files includes: 1) ASM and COE files corresponding to the testing scenario, 2) CPU project directory (to avoid the compressed file size being too large, only keep the. xpr file and the. srcs and. ip_user_files folders)。
      - NOTE: During the defense, you need to download the project you submitted from BlackBoard and generate a bitstream file on site under the supervision of the inspector. Please test your submission in advance to ensure that a bitstream that can pass the test can be generated correctly during the defense. If individual test cases fail on-site debugging, the score for application functionality will be multiplied by 0.9. If it is completely impossible to board, please resubmit the code after passing the test and convert the score according to the delay coefficient. And reschedule the defense time.
      - The course team will conduct plagiarism checks on all Verilog and ASM codes after the defense
    - The name format of the compressed file is: c defense time_group member name list:
      - For example: c16A_B_C (where c16 represents the code submission for the in class defense on week 16th, and A, B, and C are the names of three teammates)
  - **2nd submission: documents and vedios** (**Before 12:00 AM on next Monday of the week of defense**)
    - document (format: pdf) , document name: **d**_Defense Time_ group member name list. If there is no video, submit the document is enought.
    - vedio (Only record the bonus section, groups that have not implemented the bonus do not need to submit it) :
      - Please compress the documents and videos into a folder and submit them. The name of the compressed file should be: dv_defense time _group member name list

Scoring instructions (1)

- Scoring is based on code specifications (structured design, variable naming, code specifications, comments), documentation, and functional acceptance demonstrations。
- The project score consists of two parts: basic score (100 points) and bonus (15 points). If the score exceeds 100, the overflow part will be proportionally included in the overall evaluation.
  - basic part: basic function(70) + code specifications (3) + documentation (12) +Defense score(15,this part is based on individual defense performance and is not affected by contribution ratio)
    - basic function:
      - 1) Implement a single cycle CPU for textbooks (supporting RISC-VI's lw, sw, beq, add, sub, and, or), requiring submodules to pass OJ testing. (10)
      - 2) Based on the implementation of 1), expand the functionality and instruction set of a single cycle CPU, and pass the on board testing (60)
        - In the CPU code tested on the board, for the specified port of the specified module (see Appendix xx for specific module and port names), it is required to be consistent with the code submitted for testing on OJ
        - On board testing passed Basic Scenario 1 and Basic Scenario 2
        - Use I/O devices as required: buttons (function buttons such as data confirmation), dial switches (data input), LEDs (operand display), and seven segment tubes (result display). Considering the complexity of testing, basic user experience needs to be considered for data input and output. Otherwise, points will be deducted at discretion.。
  - bonus part: 15 points
    - **Note: the implementation of bonus should include corresponding code, demonstrations, documentation, and videos;**
    - The document should explain the implementation of the functions related to bonus (including the implementation mechanism, test cases, and test results). The video only requires recording a demonstration of the functions of bonus
    - **If there is a missing document or video corresponding to the bonus, the bonus score will be discounted by 60%.**

      For example, if the bonus function scores is 10 points and there is a lack of qualified videos or documents, the total score of the bonus is 10 * 0.6=6 points。
  - Additional note: If the testing on board is not valiable, the total score of the project will be multiplied by (0.3~0.5) depending on the situation.

Scoring instructions (2)

- **bonus part【max: 15】** including but not limited to following items:
  - 1) Implement support for complex I/O device (such as VGA , keyboard , etc.)【max: 2】
    - NOTE: In this course, only access to complex I/O device achieved through software and hardware collaboration is supported (i.e. accessing related complex I/O device either through corresponding instructions or through corresponding address information in instructions, rather than just using hardware control)。
    - This course focuses more on CPU architecture optimization or application scenario implementation, so the implementation effect of VGA, keyboard, or other I/O is not considered as a bonus score
  - 2) Implement only do “program device” on the FPGA chips once, and switch between multiple testing scenarios through UART interface【max: 2】
  - 3) A single cycle CPU based on RISC-V32I ISA implementation introduced in the experimental class, implementing new design ideas (such as pipeline)【6-12】
    - NOTE: It is necessary to be able to run a code snippet that contains at least one data wizard correctly. It is required to modify the assembly code used for testing on site and pass the test. (Please refer to the subsequent release documentation for basic test cases)
  - 4) To implement the lui, aupic, and ecall in ISA for existing RISC-V32I, test cases must be provided simultaneously。【1-6】
  - 5) Implementation or application of CPU based software and hardware collaboration【2-10】
    - In the toolchain of CPU software and hardware collaboration, self created small tools such as extended assembly tools that support the landing of auipc instructions on the current CPU, tools that can match and generate COE files with adjustable ROM/RAM size, hardware implementations or communication tools with adjustable UART speed, etc。
    - Software applications that can communicate and cooperate with the CPU, such as graphics processing, sound processing, upgraded game controllers, etc。
  - Note: If implementing a LoongArch instruction based CPU, there is no need to implement RISC-V CPU again (shifting the functional score and bonus score points to LA instructions), and multiply the bonus score by a coefficient of 1.05-1.1 based on the completion status (but the maximum bonus score does not exceed 15 points). Please note that the relevant documents must compare the differences between the implementation of this architecture and the RISC-V implementation introduced in class, otherwise the relevant bonus will be 0 points。

Document requirements (1-Basic function related documents)

- Developer description: Student ID, name, job responsibilities, and contribution percentage of each member.
- Development plan schedule and implementation status, version modification records (optional)
- CPU Architecture Design Description
  - CPU features:
    - ISA (Including all instructions (instruction name, corresponding encoding, usage method)) , reference ISA, the updates or optimizations made in this assignment based on the reference ISA; Registers (bit width and number) and other information; Support for exception handling。
    - CPU clk、CPI, Is it 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)。
    - Addressing space design: whether it belongs to the von Neumann structure or the Harvard structure; Addressing unit, size of instruction space and data space, base address of stack space
    - Support for IO device: Use separate instructions to access the IO device (and corresponding instructions) or MMIO (and corresponding addresses of related IO), and use polling or interrupt methods to access IO。
  - CPU interface: clk、reset、uart interface、description about other normal IO interface。
  - CPU internal structure
    - interface connection diagram of internal sub modules of CPU
    - Design Description of CPU Internal Submodules (Port Specifications and Function Description of Submodules)
- usage instructions about the system: Develop input and output operation instructions related to system operations on the board. (such as resetting the input device used, how to achieve reset; CPU working mode switch button and how to achieve mode selection; observation area of output signal, corresponding relationship with output data, etc.)
- Self testing instructions: List the testing method (simulation, on board), testing type (unit, integration), test case description (except for this article and OJ), and test results (pass or fail) in a table format; And the final test conclusion。
- Problem and Summary: Problems encountered during the development process, reflections, and summaries.
- Attention: The above content is all scoring points. Please complete the document according to the requirements. Additionally, the document does not require lengthy discussion and could be used in Chinese.

Document requirements
(2- for documents and videos related to bonus)

- Document requirements related to bonus
  - Please place explanations related to bonus in the latter part of the basic functional documentation.
  - Design description of bonus corresponding function
    - Design ideas and their relationship with surrounding modules
    - Core code and necessary descriptions
  - Test descriptions：testing scenario, test cases, test results。
  - Problem and Summary: Problems encountered during the development process of the bonus feature, reflections, and summaries.

- Vedio requirements related to bonus
  - The video should provide a complete introduction to this major assignment (including group members, overall functions, especially bonus related function points)
  - The main content should be: Introduction to the design concept, functional demonstration of bonus

Final defense requirements

- Preparation for defense:
  - 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 “program device”, and questions need to be answered by according to the related code. The two computers are convenient for synchronous testing).
- Final defense includes:
  - The demonstration and Q&A sessions require all team members to be present and answer questions.
  - Require on-site modification of assembly code according to demonstration requirements, complete the entire process of assembly, program distribution, and testing.
  - NOTE: During the defense, you need to download the project you submitted from BlackBoard and generate a bitstream file on site under the supervision of the inspector. Please test your submission in advance to ensure that a bitstream that can pass the test can be generated correctly during the defense. If individual test cases fail on-site debugging, the score for application functionality will be multiplied by 0.9. If it is completely impossible to board, please resubmit the code after passing the test and convert the score according to the delay coefficient. And reschedule the defense time.
- During the demonstration process, it is necessary to complete the onboard testing of the CPU (Minisys/EGO1 development board) as required.
  - Basic testing scenes on CPU (Please refer to the specific content on the following page)
  - Expansion functions of CPU (refer to the "bonus functions" section in P5)

Basic Test Scenario 1 (relatively simple, accounting for 30%)

Use the dial switches on the development board for input, including 3 dial switches (x2,.. X0) for inputting test case numbers, 8 dial switches (sw7,.. Sw0) for inputting test data, and using LED lights or 7-segment tubes for output (If the number of LED lights is not enough, the output will be displayed alternately on the LED lights, with each part displayed for 2 seconds.) (Note that the high bit of the DIP switch and LED will be unified on the left, and the low bit will be unified on the right.)
NOTE 1: If there are multiple input data (such as a, b), please enter a first, press the confirm button, and then enter b
NOTE 2: Use LED to output the binary form of the operand, and use 7-segment tube (or VGA) to output the hexadecimal form of the operation result.

Scenario 1. testcase number	testcase description (in testcase 3'b011-3'b111, both a and b are in binary complement form)
3‘b000	Input test number a, input test number b, and display the 8-bit binary format of a and b on the output device (LED)
3‘b001	Input the test number **a**, place it in a register by instruction **lb**, display the value of the 32-bit register in hexadecimal format on the output device (7-segment tubes or VGA), and save the number to memory (in the 3'b011-3'b111 test case, the value of a will be read from the memory unit using the lw instruction for comparison)
3‘b010	Input the test number **b**, place it in a register by instruction **lbu**, display the value of the 32-bit register in hexadecimal format on the output device (7-segment tubes or VGA), and save the number to memory (in the 3'b011-3'b111 test case, the value of b will be read from the memory unit using the lw instruction for comparison)
3‘b011	Compare test number **a** and test number **b** (from testcase 3’b001 and testcase 3’b010) using instruction **beq**. If the relationship is true, light up the LED, but if the relationship is not true, turn off the LED
3‘b100	Compare test number **a** and test number **b** (from testcase 3’b001 and testcase 3’b010) using instruction **blt**. If the relationship is true, light up the LED, but if the relationship is not true, turn off the LED
3‘b101	Compare test number **a** and test number **b** (from testcase 3’b001 and testcase 3’b010) using instruction **bge**. If the relationship is true, light up the LED, but if the relationship is not true, turn off the LED
3‘b110	Compare test number **a** and test number **b** (from testcase 3’b001 and testcase 3’b010) using instruction **bltu**. If the relationship is true, light up the LED, but if the relationship is not true, turn off the LED
3‘b111	Compare test number **a** and test number **b** (from testcase 3’b001 and testcase 3’b010) using instruction **bgeu**. If the relationship is true, light up the LED, but if the relationship is not true, turn off the LED

Basic Test Scenario 2 (relatively complex, accounting for 70%)

- Use the dial switches on the development board for input, including 3 dial switches (x2,.. X0) for inputting test case numbers, 8 dial switches (sw7,.. Sw0) for inputting test data, and using LED lights or 7-segment tubes for output (If the number of LED lights is not enough, the output will be displayed alternately on the LED lights, with each part displayed for 2 seconds.) (Note that the high bit of the DIP switch and LED will be unified on the left, and the low bit will be unified on the right.)
- NOTE 2: Use LED to output the binary form of the operand, and use 7-segment tube (or VGA) to output the hexadecimal form of the operation result

Scenario 2. testcase number	testcase description (The input data for testcase 3 'b001-3' b011 is a signed number)
3‘b000	Input an 8-bit number, calculate and output the number of leading zeros(the number of leading zeros of 8’b00010000 is 3)
3‘b001	Input a 16 bit IEEE754 encoded half word floating-point number, round it up, and output the rounded result
3‘b010	Input a 16 bit IEEE754 encoded half word floating-point number, round it down, and output the rounded result
3‘b011	Input a 16 bit IEEE754 encoded half word floating-point number, rounding it, and output the rounded result
3‘b100	Input numbers a and b(each of them is 8-bit) , perform addition operations on a and b. If the sum exceeds 8 bits, remove the high bits and add them up to the sum. then inverting the sum, output the result
3‘b101	Input 12 bit data in little endian mode from the dial switch and present it on the output device in big endian mode
3‘b110	Calculate the number of Fibonacci numbers which are smaller than the input data in a recursive manner, record the number of times the stack is pushed and pushed, and display the sum of the push and push times on the output device
3‘b111	Calculate the number of Fibonacci numbers which are smaller than the input data in a recursive manner, record the push and pop data, display the push data on the output device(each push data display for 2-3 seconds (note that here not focus on the push and pop of the value of ra regisger)

总体说明 (重要!)

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**一周内完成基本测试**，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，不到场则自动减少组队人数，贡献比按实际人数安排。**
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）。
(贡献比最小值和最大值不超过10%：即二人组极限比例是45:55，三人组极限比例是30:30:40)

个人Project总评 = 团队得分 * 团队人数 * 个人贡献百分比 + 个人答辩表现分(15)
团队得分 = 功能验收分(70) * **推迟/提前系数(0.6~1.05)** + 代码规范分(3) + 文档(12) + bonus(15)

答辩 说明	代码提交	文档（视频）提交	提前/推迟系数（乘以功能验收分）
中期答辩（13周实验课）	不涉及	不涉及	不涉及
提前答辩（15周实验课）	15周周一中午12:00前	16周，周一中午12:00前	1.05
正常答辩（16周实验课）	16周周一中午12:00前	17周，周一中午12:00前	1
推迟答辩（所有延迟小组提交后统一安排）	迟一天扣5%系数	17周，周一中午12:00前	16周周二0.9，周三0.85，周四0.8，周五0.75，周六0.7，周日0.65，17周周一0.6，代码提交截止时间均为当天中午12:00

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计算“功能验收分”。
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。**
个人总评说明：如三人组贡献比为1:1:1，15周答辩，则最高分可得（70*1.05+3+12+15)*3*0.33+15 = 118.5

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（答辩当周的周一中午12：00之前提交bb站点）**
    - 提交内容：测试场景对应的asm以及coe文件，CPU project目录 (为避免压缩包尺寸过大，仅保留.xpr文件以及**.srcs**和**.ip_user_files**文件夹即可)。**答辩时需从bb下载你提交的project，在inspector监督下现场生成bitstream文件，请提前测试好你的提交内容，确保答辩时能正确生成可通过测试的bitstream。个别用例测试不通过可现场调试，届时对应用例功能得分将乘以0.9。如果完全无法上板，请测试通过后重新提交代码，按照推迟系数折算分数。并重新预约答辩时间。**
    - 注意，课程组将于答辩后对所有verilog、asm代码进行查重
    - 压缩包的名字格式为：　　**c答辩时间_小组成员姓名列表**

    比如： c160156_A_B_C (其中c160156表示16周周一56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中午12：00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（只录bonus部分，没有实现bonus的小组无需提交）：
      - 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（100分）+ bonus（15分），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(70) + 代码规范（3）+ 文档（12）+答辩分数(15,此分数为个人答辩表现分，不受贡献比影响)
    - 基本功能：
      - 1) 实现课本的单周期CPU（支持 RISC-VI的 lw,sw,beq, add, sub, and, or)，要求子模块通过OJ测试. （10）
      - 2) 基于1）实现的单周期CPU做功能和指令集拓展，能够实现上板测试 （60）
        - 上板测试的CPU代码中，对于指定模块的指定端口（具体模块和端口名参见附录xx），要求与OJ上测试提交的代码一致
        - 测试通过基本场景1、基本场景2
        - 按要求使用外设：按键(功能按钮如数据确认等)、拨码开关(数据输入)、**led(操作数展示)**和七段数码显示管(结果展示)，考虑到测试复杂度，数据输入、展示需考虑基本的用户体验，否则将酌情扣分。
  - bonus：15分
    - **请注意：bonus的实现应包含相应的代码、演示、文档及视频；**
    - **其中文档应就bonus相关的功能实现进行说明（包括**实现机制、测试用例以及测试结果的说明**，视频仅要求录制bonus的功能演示）**
    - **如缺少bonus对应的文档或者视频，bonus分数打6折。**

      比如bonus功能得分10分，缺少合格的视频或者缺少文档，则 bonus 的总分 = 10*0.6 = 6分。
  - 补充说明：如果不能上板测试，则根据情况，项目总得分*(0.3~0.5)

评分说明（2）

- **bonus 功能【max: 15】** 包括但不限于：
  - 1）实现对复杂外设接口的支持（如VGA接口、键盘接口等）【max: 2】
    - 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)。
    - 本课程bonus更偏重于CPU架构优化或应用场景实现，因此VGA或键盘或其他外设实现效果不作为bonus分数高低考量
  - 2）实现只烧写一次FPGA芯片，可通过uart接口实现多个测试场景之间的切换【max: 2】
  - 3）基于实验课介绍的 RISC-V32I ISA 实现的单周期CPU，实现新的设计思路（如pipeline）【6-12】
    - 说明：需能够正确运行至少包含一个data hazard的代码片段，要求现场修改测试用的汇编代码并测试通过。（**基本测试用例请参考后续发布的说明文档**）
  - 4）实现现有RISC-V32I 的ISA中的 lui，aupic，ecall 需同时提供测试用例。【1-6】
  - 5）基于CPU的软硬件协同的实现或应用【2-10】
    - CPU软硬件协同的工具链中自创小工具，如支持auipc指令在当前CPU上落地的扩展汇编工具，可匹配生成ROM/RAM大小可调整的coe文件创建工具，uart速率可调整的硬件实现或者通信工具等。
    - 可以与CPU进行通信并配合的软件应用，如图形处理、声音处理、升级版游戏手柄等。
  - **注意**：如果实现基于LoongArch指令CPU，则无需再实现RISC-V CPU，(功能分和bonus得分点平移到LA指令），并根据完成情况在bonus得分基础上乘以1.05~1.1的系数（但bonus最高不超过15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RISC-V实现之间的**差异，否则相关bonus为0分。**

文档要求（1-基本分的相关文档）

- 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。
- 开发计划日程安排和实施情况，版本修改记录（可选)
- 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。
    - CPU时钟、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）。
    - 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，栈空间的基地址。
    - 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接口：时钟、复位、uart接口、其他常用IO接口说明。
  - 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（子模块端口规格及功能说明）
- 系统上板使用说明：开发板上与系统操作相关输入、输出操作说明。（如复位使用的输入设备、如何实现复位；CPU工作模式切换的按键及如何实现模式选择；输出信号的观测区域，与输出数据的对应关系等）
- 自测试说明：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（除本文及OJ以外的用例）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。
- 注意：以上内容均为评分点，请大家按照要求完成文档，另外文档不需要长篇大论，可以使用中文。

文档要求（2-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

最终答辩要求

- 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。
  - **提醒：答辩时需从bb下载小组提交的project，在inspector监督下现场生成bitstream文件，请提前测试好你的提交内容，确保答辩时能正确生成可通过测试的bitstream。个别用例测试不通过可现场调试，届时对应用例功能得分将乘以0.9。如果完全无法上板，请测试通过后重新提交代码，按照推迟系数折算分数。并重新预约答辩时间。**
- 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中“**bonus 功能**”部分）

基本测试场景1（实现相对简单，占测试场景分值的 30%）

使用开发板上的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.（如果led灯的数目不够，则将输出在led灯上交替展示，每个部分展示停留2秒）（备注，拨码开关和led的高位统一在左侧，低位统一在右侧）

说明1：如果输入数据有多个（比如a，b），请先输入a，按确认键，再输入b

说明2：用led输出操作数的二进制形式，使用数码管（或者VGA）输出运算结果的十六进制形式。

场景1.用例编号	用例描述（用例3’b011-3’b111中，a，b均为二进制补码形式）
3‘b000	输入测试数a，输入测试数b，在输出设备（led）上展示8bit的a和b的值
3‘b001	输入测试数a，以lb的方式放入某个寄存器，将该32位的寄存器的值以十六进制的方式展示在输出设备上（数码管或者VGA)，并将该数保存到memory中 （在3’b011-3’b111用例中，将通过lw 指令从该memory单元中读取a的值进行比较）
3‘b010	输入测试数b，以 lbu 的方式存入某个寄存器，将该32位寄存器的值以十六进制的方式展示在输出设备上（数码管或者VGA)，并将该数保存到memory中（在3’b011-3’b111用例中，将通过lw 指令从该memory单元中读取a的值进行比较）
3‘b011	用 beq 比较 测试数 a 和 测试数 b（来自于用例1和用例2），如果关系成立，点亮led，关系不成立，led熄灭
3‘b100	用 blt 比较 测试数 a 和 测试数 b（来自于用例1和用例2），如果关系成立，点亮led，关系不成立，led熄灭
3‘b101	用 bge 比较 测试数 a 和 测试数 b（来自于用例1和用例2），如果关系成立，点亮led，关系不成立，led熄灭
3‘b110	用 bltu 比较 测试数 a 和 测试数 b（来自于用例1和用例2），如果关系成立，点亮led，关系不成立，led熄灭
3‘b111	用 bgeu 比较 测试数 a 和 测试数 b（来自于用例1和用例2），如果关系成立，点亮led，关系不成立，led熄灭

基本测试场景2（实现相对复杂，占测试场景分值的 70%）

- 使用开发板上的拨码开关做输入，其中3个拨码开关（x3-x0）用于测试用例的编号输入，8~16个拨码开关用于做测试数据的输入（备注，拨码开关、led以及数码管的高位统一在左侧，低位统一在右侧）
- **说明2：用led输出操作数的二进制形式，使用数码管（或者VGA）输出运算结果的十六进制形式。**

场景2.用例编号	用例描述（用例3‘b001-3’b011的输入数据为有符号数）
3‘b000	输入一个8bit的数，计算并输出其前导零的个数
3‘b001	输入16bit位宽的IEEE754编码的半字浮点数，对其进行向上取整，输出取整后的结果
3‘b010	输入16bit位宽的IEEE754编码的半字浮点数，对其进行向下取整，输出取整后的结果
3‘b011	输入16bit位宽的IEEE754编码的半字浮点数，对其进行四舍五入取整，输出取整后的结果
3‘b100	分两次输入两个8bit的数a和b，对a，b做加法运算，如果相加和超过8bit，将高位取出，累加到相加和中，对相加和取反后输出
3‘b101	输入12bit的数据，以小端模式从拨码开关输入，以大端的方式呈现在输出设备上
3‘b110	以递归的方式计算小于输入数据的斐波拉契数字的数目，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‘b111	以递归的方式计算小于输入数据的斐波拉契数字的数目，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）