蜂鸟 E203 处理器内核的第一级处理器流水线主要承担以下操作：

1. 分支预测
2. 生成 PC
3. 根据 PC 取得指令并存入 IR
4. 将相关数据传入 ex 级

1. 分支预测

对于有条件分支指令，E203 处理器内核采用静态预测，只要是向后跳转，就预测为正确。在具体实现中通过判断立即数字段的符号位是否为正来判断是否为向后跳转。对于无条件分支指令，jal 指令直接跳转即可，但 jalr 因涉及到读取寄存器情况则略微复杂。首先，因为 0 号寄存器即为常数 0 因此无需读寄存器。其次，因为 RISC-V 规定 jalr 的目标寄存器索引若为 1 或者 5，则视为函数调用或者返回行为，而利用 1 号寄存器进行函数调用和返回更尤为频繁，因此在具体实现中将 1 号寄存器旁路至分支预测单元，若读取除 0 和 1 号寄存器则还需要等待一个周期。分支预测操作的主要承担模块是 bpu，为 bpu 提供判断依据(解码指令，指示是何种指令，并提供相应操作数)的模块为 minidec。minidec 复用自 ex 级的 decoder。

2. 生成 pc

pc 的产生共有 5 种情况。第一种情况是发生了跳转指令，计算目标地址的两个操作数由 bpu 给出。第二种情况是发生了来自 top 的复位请求，计算目标地址的第一个操作数来自 top 的默认复位地址，第二个操作数为全 0。复位信号到来后先经过锁存，再经一个寄存器同步时序后方可使用。锁存复位信号的寄存器为不带写使能且默认复位值为 1 的寄存器，将复位信号接入到该寄存器的复位端口后，只要复位信号到来，该寄存器就会因其默认复位值为 1 而变成高电平，从而达到锁存复位信号的效果。第三种情况是顺序取指，计算目标地址的第一个操作数为当前 pc 寄存器的 pc，第二个操作数为顺序自增的偏移量。顺序自增的偏移量根据指令的字节数而改变，2 字节的 C 扩展指令偏移量是 2，4 字节的指令偏移量是 4，指令的字节数由 minidec 的 rv32 指出。第四种情况是发生了流水线冲刷，当产生异常、中断或者跳转预测错误时都会发生流水线冲刷，需要冲刷掉当前指令，并把流水线冲刷的目标 pc 写入 pc 寄存器。流水线冲刷 pc 可能是正确的跳转目标地址，也可能是异常或者中断的处理函数地址。流水线冲刷的请求来自 ex 级，但是到达 if 级的时候可能时机不对，到达时已经取完 IR 或者正在取 IR，此时流水线冲刷信号无法冲刷掉当前的指令，因此就出现了第四种情况延迟的流水线冲刷请求。

生成 PC 并生成取指令请求的单元是 fetch，其执行流程依次为更新 PC、生成取指令的请求、更新 PC 寄存器和接收指令并更新 IR 和其相关寄存器。在更新 PC 阶段有流水线冲刷请求到来时可以轻松的冲刷掉当前指令，因为 PC 寄存器在存在流水线冲刷请求的时候也会写使能，从而使得流水线冲刷请求的目标 pc 写入 pc 寄存器，从而达到冲刷指令和写入起始地址的操作。但在生成取指令的请求之后的阶段请求到来时却无法冲刷掉当前指令，只能将 PC 寄存器冲刷掉，并且由于已经过了取指令的时序，因此只能等到下一个周期执行 PC 寄存器中的流水线冲刷目标 PC，所以流水线冲刷信号也将被锁存住，用于在下个周期指示 pc_next 选择 pc_r(此时 PC 寄存器存储的是流水线冲刷目标 PC)。总之，当 fetch 处于更新 PC 和生成取指令的请求的阶段，流水线冲刷可顺利冲刷掉当前指令并执行流水线冲刷目标 PC，当处于其他阶段，流水线冲刷无法顺利冲刷当前指令，仅可冲刷掉 PC 寄存器并更换为流水线冲刷目标 PC，因此需要等待下一个周期才可执行流水线冲刷目标 PC。具体实现中以~ifu_req_hsked 表示当前 fetch 不处于更新 PC 和生成

取指令的请求的阶段。ifu_req_hsked 表示 fetch 与 ift2icb 的 cmd 信道握手成功即成功生成读取指令请求，正在传输 pc，而更新 pc 阶段与其并行执行。以 dly_flush_xxx 表示锁存延迟的流水线冲刷信号的寄存器，该寄存器的输出信号 dly_pipe_flush_req，将用于产生 PC 的第四种情况的选择信号，选择当前 PC 寄存器的内容(pc_r)做为读取指令的 PC。

第五种情况是 PC 顺序递增，需注意的是，由于该处理器支持 C 扩展的 16 位指令集，因此偏移量可能为 2 也可能为 4。每条指令的长度由低两位指出，在 minidec 中可解码得出信号 rv32，用于指示该 pc 为 16 位还是 32 位，因此将 minidec 的 rv32 输入至 fetch 中用于判断偏移量是 2 还是 4。

此外需要特殊注意一下 PC 寄存器，根据 pc 寄存器的使能信号 pc_ena = ifu_req_hsked | pipe_flush_hsked 可知，在不考虑流水线冲刷的情况下，当 ift2icb 与 fetch 的 cmd 通道握手后才写使能，因此写入 PC 寄存器应发生在下一个 clk 的上升沿，在此 clk 到来之前，pc_next 表示当前将要执行的 PC，PC_r 表示上一个周期执行的指令。

3. 根据 PC 取得指令并存入 IR

该步骤应拆分为 fetch 向 ift2icb 发送请求读取指令信号和 PC 等相关数据、根据 pc 读取指令、ift2icb 向 fetch 写回指令以及更新 IR 相关寄存器四部分，仅第三部分发生在 ift2icb 单元，其他均大部分发生在 fetch 单元，以下将一一总结出来，并且以下四部分在处理器中是按顺序执行的。

一，fetch 向 ift2icb 发送请求读取指令信号和 PC 等相关数据

该部分用到了 ICB 协议，发生在 ICB 协议的 cmd 通道，先由 fetch 向 ift2icb 发送读写请求信号，待 ift2icb 回传读写请求准许信号则 cmd 通道握手成功可以发送数据，握手成功后即可向 ift2icb 传输数据，数据包括待读指令的 PC、该 pc 的位数、是否为顺序递增取指操作和上一条指令的 pc。以上全部操作均发生在一个 clk 内，除握手信号以外是无先后顺序的，读写准许信号到来之前是无法向后推进的。在发送读写请求信号之前，还应判断是否可以产生新的读写请求，应满足以下两个条件。第一个条件是无 bpu_wait 和 ifu_halt_req(处理器暂停工作信号)，且复位请求和流水线冲刷请求均可触发 fetch 产生读写请求信号。第二个条件是前一个读写请求完成 rsp 通道的握手，完成 rsp 通道的握手代表着上一个读写请求信号执行完毕，ift2icb 单元才可读取下一个 pc。

二，根据 pc 读取指令

三，ift2icb 向 fetch 写回指令

该部分属于第一部分建立的 icb 通信总线的 rsp 通道，当指令从存储器取出后，由 ift2icb 向 fetch 发送读写请求反馈信号，待 fetch 回传读写请求反馈准许信号则 rsp 通道握手成功可以发送数据，数据包括访存错误位和根据 PC 取来的指令。在回传读写反馈请求准许信号前，应满足 IR 寄存器可被写入，共两种情况下 IR 可被写入。一是发生流水线冲刷，二是 IR 为空或者 IR 即将被清除且无 bpu_wait 发生。

四，更新 IR 相关寄存器

当读取到指令后，需将指令写入 IR 寄存器中，其他需输出至 ex 级的数据也应同 IR 寄存器一起更新。需输出至 ex 级的数据包括指令、该指令对应的 pc、指令非对齐、访存错误、寄存器端口 1 的索引、寄存器端口 2 的索引、表示分支预测是否为真和是否发生乘除法 b2b，以上信号均需提前写入相应的寄存器中。