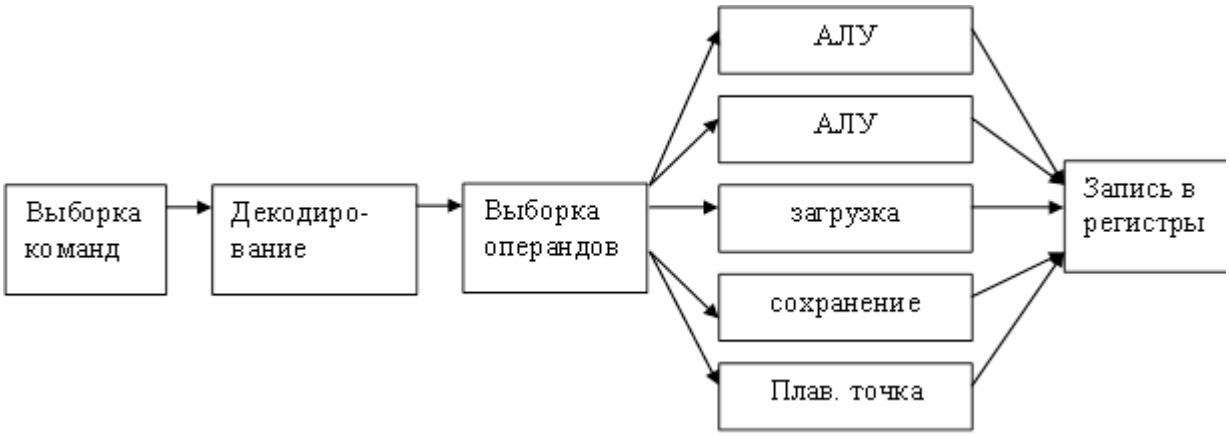
Низкоуровневый параллелизм достигается, в частности, вызовом нескольких команд за один тактовый цикл. Процессоры, в которых реализуется этот принцип:

- **суперскалярные** (способны за один тактовый цикл могут вызывать несколько команд (обычно от 2 до 6), что определяется как аппаратной реализацией, так и последовательностью команд)

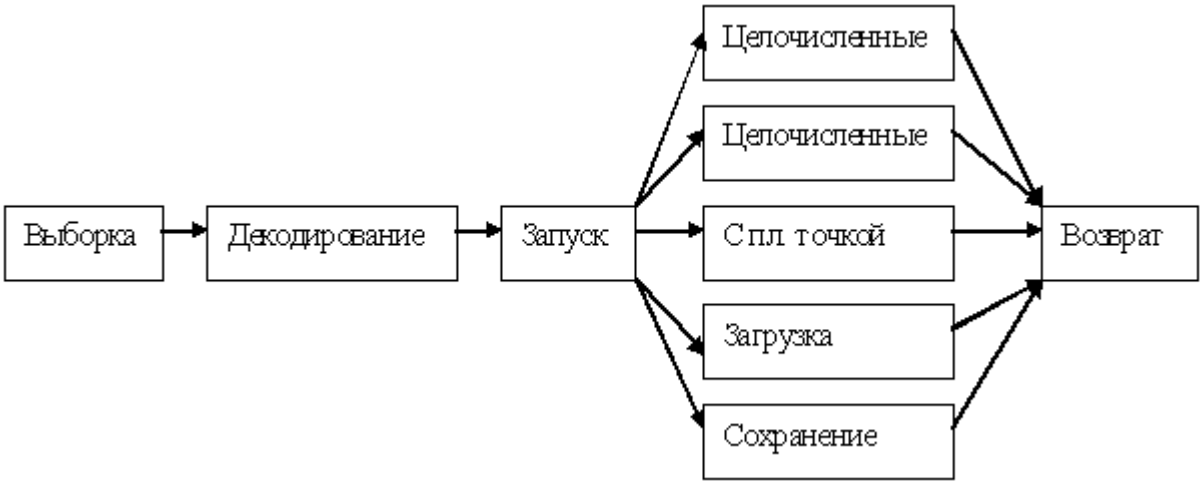


- **VLIW** (Very Long Instruction Word – процессоры со сверхдлинным словом).

В современный VLIW-системах должен быть предусмотрен механизм маркировки связи команд с тем, чтобы процессор мог выбрать и запустить связку. Задача по подготовке и завершению связок выполняется компилятором. Таким образом в VLIW-системах решение проблем совместимости переносится со стадии исполнения на стадию компиляции.

Преимущества этого:

1. Упрощается аппаратное обеспечение;
2. Поскольку на стадии компиляции нет жестких временных ограничений, то поиск и составление связок может быть выполнен более качественно.



Проблема: если при запросе к памяти требуемые данные не обнаружены в КЭШах первого или второго уровней, то на загрузку слова в КЭШ уходит длительное время, во время которого конвейер простаивает.

Решение: внутрипроцессорная многопоточность (on-chip multithreading). Позволяет процессору одновременно управлять несколькими программными потоками, тем самым маскировать простои.

Общая идея: если программный поток 1 блокируется, процессор может обеспечить полную загрузку аппаратуры, запустив программный поток 2.

Способы реализации:

Мелкомодульная многопоточность (fine-grained multithreading). Простой маскируется путем исполнения потоков «по кругу», т.е. в смежных циклах запускаются по циклу. Поскольку время обращения в память составляет 2 цикла, то при трех потоках гарантируется полная загрузка аппаратуры. Т.е при простое в n циклов необходимо иметь n+1 поток.

Поскольку разные программные потоки никак друг с другом не связаны, каждому нужен свой набор регистров. Следовательно, максимальное число одновременно исполняемых программных потоков определяется а стадии разработки микросхемы.

Поток 1	A1	A2			A3	A4	A5			A6	A7	A8
Поток 2	B1			B2			B3	B4	B5	B6	B7	B8
Поток 3	C1	C2	C3	C4			C5	C6			C7	C8
Мелкомодульная	A1	B1	C1	A2	B2	C2	A3	B3	C3	A4	B4	C4
Крупномодульная	A1	A2		B1		C1	C2	C3	C4		A3	A4

Крупномодульная многопоточность (coarse-grained multithreading). В этом случае программный поток А выполняется до тех пор, пока не возникнет простой. При этом теряется один цикл. Затем происходит переключение на другую задачу. При переключении потока теряется один цикл. Кажется, что крупномодульная многопоточность менее эффективна. Однако это справедливо для большого числа доступных потоков (n+1), а при малом числе потоков крупномодульная многопоточность оптимальна.

В суперскалярных процессорах используется **синхронная многопоточность** (simultaneous multithreading). Эта методика представляет собой усовершенствованный вариант крупномодульной многопоточности, где каждый программный поток может запускать по две команды за такт. В случае простоя с целью обеспечения полной загрузки процессора запускаются команды следующего потока. При синхронной многопоточности обеспечивается полная загрузка функциональных блоков. В случае невозможности запуска команды из-за занятости функционального блока выбирается команда из другого потока.