Въпреки че концепцията за суперскаларния подход е характерен за RISC процесорите, същите суперскаларни принципи могат да се внедрят и при CISC машините. Естествено че най-подходящ пример е Pentium4. Еволюцията на суперскаларните концепции в фамилията Интел е интерсна за разглеждане. 386 е традиционен CISC неконвейрна архитектура. 486 въвежда първия конвейрен x86 процесор, като намалява средно латентността от изпълнението на целочислените операции от 2 или 4 такта на 1, но все още е било ограничено изпълнението от подаването по една инстр на всеки такт без никакви суперскаларни елементи. Оригиналният Pentium има непретенциозен суперскаларен компонент , състоящ се от използването на 2 отделни изпълняващи у-ва. PentiumPro въвежда напълно оформена суперскаларна архитектура. Следващите модели експолатират новите тенденции при суперскаларните процесори. На фиг. е показана обща блок диаграма на архитектурата. На следващият слайт е показана същата архитектура по начин по-удобен за разглеждането на конвейрната организация. Операциите могат да се обобщат по следния начин:

(1) Процесорът извлича инструкциите от паметта последователно от статичната програма;
(2) Всяка инструкция се декодира на 1 или повече RISC инструкции с фиксирана дължина, т.нар. микро-операции или микро-опс.
(3) Процесорът изпълнява микрооперациите върху суперскаларната организация, по този начин , микроопс могат да бъдат изпълнени неподредено(out-order).
(4) Процесорът записва резултата(фаза на отегляне) на всяка микро-опс в регистровото множество последователно спрямо оригиналния програмния поток.

В р-т от направените по-горе изводи Pentium 4 архитектурата се състои от външна CISC обвивка с RISC ядрото. Инструкциите на вътрешно RISC ядро преминават през конвейра за 20 фази. В някои случаи микроопс се нуждаят от няколко пъти повтарянето на 20те фази. Това направо звучи различно от 5 фазното изпълнение което ние разглеждаме на упражненията. Следва последователно разглеждане на операциите на конвейра показано на следващите слайтове.

Фаза а) **generation of micro-ops**
Има пре-процесор: front-end,който е с политика in-order, и е извън конвейрната организация. Този front-end зарежда L1 инстр кеш като извиква трейс кеша и оттук нататък вече е конвейрната организация. Обикновенно процесора работи с трейс кеша и само ако има miss cache този препроцесор извлича инструкциите и ги записва в трейс кеша това е първата фаза. С помощта на BTB и I-TLB устройството за извличане и декодиране извлича машинните инстр от L2 кеш за всеки такт по 64B. Инструкциите по дефоулт се извличат последователно така, че всяко извличане на кеш линия включва указание че следващата инструкция да бъде извлечена. ILTB превежда линейния инструкционен адрес, зададен от ФА за да достъпе кеш L2. Статичната логика за предсказване в препроцесора е BTB и се използва за да открие кои инструкции следват да се извлекат.
Веднъж като са извлечени инстр, у-вото за извличане/декодиране сканира байтивете с които се задава инструкцията; това е необходима операция, защото дължината е променлива на x86. Декодера преобразува всяка машинна инструкция от 1 на 4 микроопс, всяка от които е 118-б RISC инструкция. В сравнение с простите RISC машини при които дължината на инструкцията е 32б. По-дългите микроопс се нуждаят от изпълнението на по-сложни пентиум операции. Микроопс са по-лесни за управление отколкото оригиналните инструкции от които те идват.
Генерираните микроопс се записват в трейс кеша.

(Фаза b)**Trace Cache next Instr Pointer** – Първите две конвейрни фази са свързани с избор на инструкции в трейс кеша и включване на механизъм на предсказване на преход. Пентиум 4 използва динамична стратегия за предсказване на преход, която се основава на анализ историята на досега изпълнените инструкции за преход. BTB съдържа тази информация. Дали една бранч инстр се е предсказала и е била подадена към инструкционния поток, то BTB се отбелязва полето за предсказани правилно инстр. Ако тази инстр вече е била предсказана, т.е. съществува в BTB тогава инст у-во се направлява от историята в тази таблица. Ако преходът е предсказан тогава адреса дестинация на бранча записан срещу инстр се взима за да се извлече бранч таргет инструкцията. Веднъж като инстр се изпълни полето история се подновява с резултата от бранч инстр. В противен случай ако я няма в BTB адреса на тази инструкция се зарежда в полето на таблицата.
В пентиум са използвали проста 2битова историческа схема. Но за разлика от P4 пентиум има само 5 фази на конвейра. Затова тук се използва по-усложнена схема за предсказване с повече битове. При P4 BTB е организирана като 4посочет групово-асоциативен кеш с 512 линии. Всяка инстр използва адр на бранча като таг. Полето за история е 4бита.За разлика при повечете CC п-ри и оригиналния Пентиум. Алгоритъма по който се анализира BTB или тези 4 бита е известен като Yeh's algorithm. Тези който са го разработили този алгоритъм твърдят че осигурява голямо намаляване на неправилно предсказаните преходи в сравнение с 2бита на предишния.
При условният преход няма история и в BTB се предсказва като използват статични алгоритми, основани на следните правила.
Фаза 4c) **trace cache fetch** Тейс кеша взима вече декодираните микроопс от инстр декодер и асемблира ги до програмна последователност от микроопс наречена трейсове. Микроопс се извличат последователно от трейс кеша.
Някои инструкции изискват повече от 4 микроопс. Тези инструкции се прехвърлят ROM, който съдържа поредица от микроопс(обединени в една маш инстр)

Фаза 5 инструкциите отиват в преименуващия модул. Тази част на конвейра преподрежда микроопс за да им позволи да се изпълняват веднага щом има готови операнди.

Фаза 6 **Allocate** (е)заделя ресурсите необходими за изпълнението. Представя следните ф-ции:

- ако се нуждае от р-с като р-р който е зает за една от 3те инструкции, които пристигат в алокатора на всеки такт, той открадва тактове, като прекъсва п-ра;
- алокатора поставя ст-ти в ROB за една от 126 микроопс който са в процес на изпълнение;
- алокатора разпределя една от 128 интегер и FP р-ри за запис на р-та на микроопс, и разпределя един от 48 четене и 24 записа в машиния конвейр.

ROB е кръгъл буфер който може да съдържа 126 микроопс и 128 хардуерни р-ра. Микроопс влизат последователно в този буфер. Те се разпределят към изпълняващото у-во. Условието при което се подават по-нататък инстр е да са налични тяхните операнди. Оттеглянето е подредено. Микрооперациите се разпръскват към изпълняващите у-ва.
Фаза 6 **Renaming** (е) има 16 арх р-ра, които се използват за преименуване(8 fp + EAX, EBX, ECX, EDX, ESI, EDI, EBP и ESP). В тази фаза се премахват фалшивите зависимости чрез тези 16 р-ра. След като се премахнат зависимостите микроопс се подават към една от двете опашки(фиг f). Едната опашка е за операции с паметта, а другата за микроопс. Опашките са реализирани като FIFO. Но микроопс могат да се

четат от двете опашки непоследователно. Това осигурява по-голяма гъвкавост при у-вото за разпределяне.
Фаза 7 **Mikro-ops scheduling and dispatching** (g) тази фаза отговаря за подаване на микроопс за изпълнение. За да се изпълни инстр трябва устройството sheduler да следи дали са налични всички операнди. Фаза (h) у-вото за изпълнение е свободно то се подава микроопс. На всеки такт могат да се подават до 6 инструкции. Ако има повече от една инструкция чакаща едно и също изплънително у-во scheduler ги подава в последователността от опашката и тъй като дисциплината на опашките е FIFO подаването ще с политика issue in-order . Но преди изплънението инструкционния поток може да бъде преподреден от зависимостите и преходите и тогава говорим за политика out-order.

Integer and FP Execution Units

РФ-ове са източник на т.нар висящи операции – фаза (i). Фаза (j) – изп у-ва записват р-тите си в РФ-ове и работат с опернади, който се зареждат от кеш1 за данни. Отделна фаза е необходима за изчисляване на флаговете за бранч инструкциите – това става в фаза (k). А във фаза (l) става намирането на новия адрес от инст за преход. И това ще предизвика презареждане на целия конвейр отново от новият таргет адрес.