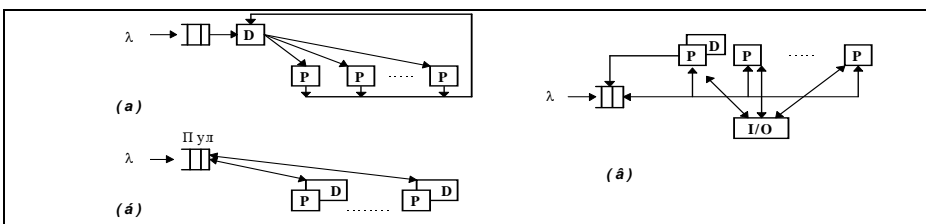


а) *централизирано диспечериране* - изпълнява се от един процесор, в чиято памет е пълната операционна система, включително диспечерната програма, а останалите процесори са изпълнители на предоставеното им работно натоварване;

б) *децентрализирано диспечериране* - копия на диспечера се съхраняват в паметта на отделните процесори и всеки самостоятелно планира своето натоварване чрез извличане на задачи от общия входен пул;

в) *комбинирано диспечериране* - всички процесори могат да изпълняват диспечерски функции (децентрализиран принцип), но един се инициализира за диспечер при конкретно конфигуриране на системата (централизиран принцип).



Механизми за диспечериране и реализация на плана, прилагани при организация на компютърната обработка в различни архитектури, са следните (допуска се комбиниране):

а) *матричен механизъм* - в паметта са дефинирани матрици, чрез които се отразяват информационните взаимовръзки между задачите;

б) *механизъм на семафорите* - всяка операция се свързва със семафор и може да се изпълни само ако той е "отворен" (подходящ за децентрализирано диспечериране при произволен план);

в) *информационен механизъм* - процесорите се натоварват в зависимост от промяната на изходната информация за работното им натоварване (приложим при много задачи с малка трудоемкост);

г) *адресен механизъм* - диспечериране на операциите в зависимост от адресите на инструкциите в паметта (поради значителната загуба на машинно време е оправдан при трудоемки операции);

д) *механизъм на директния избор* - избор по приоритет или по време за решаване на независими задачи (процеси), подредени в опашка.

Диспечерирането при единичен ресурс (псевдопаралелизъм) се състои в създаване на оптимален график (план) за изпълнение на постъпващите в КС потребителски задачи. Целта на планирането е да се подобрят системните характеристики при съществуващите ограничения. Такива характеристики са средно време за изчакване при обслужване на задача в КС (w), средно време за отговор на задача в КС (u), общо време за обработка на задачите (t).

Същността на планирането е: ако системният ресурс е свободен се планира поредна задача, която е в състояние "готовност" и чака освобождаване на заявения ресурс. За целта се създава *оптимален план* за изпълнение на реалното натоварване за даден период от време в зависимост от заложения режим на работа, наличния системен ресурс и системните ограничения, наложени от избрания критерий за ефективност. Планирането на независими процеси се основава на *планиращи алгоритми (стратегии)*, ориентирани към определена КА и избран критерий за оптималност на плана. Една група от стратегии за планиране са свързани с *трудоемкостта* на процесите. При системен ресурс от N елемента $S = \{S_1, \dots, S_N\}$ и M задачи $Z = \{Z_1, \dots, Z_M\}$, конкуриращи се при неговото използване, се дефинира *матрица на трудоемкостите* $T[N \times M] = [t_{ij}]$ за $i=1 \div M$; $j=1 \div N$, където $t_{ij} \geq 0$ определя обем памет или време за използване на ресурс S_j от задача (процес) Z_i (при $t_{ij}=0 \Rightarrow Z_i$ не заема S_j).

1.3. Особенности на диспечерирането при мултипрограмен КС с пакетна обработка

♦ *Базови методи за диспечериране FCFS/LCFS и SJF.* Прилагат се при известна трудоемкост τ_i на отделните задачи Z_i , обработвани заедно и се състоят в следното:

FCFS (First-Come-First-Served) - последователно безприоритетно обслужване на задачите, формирани (по случаен закон) т.нар. пакет;

LCFS (Last-Come-First-Served) - всяка постъпваща задача прекъзва изпълняваната в процесора такава;

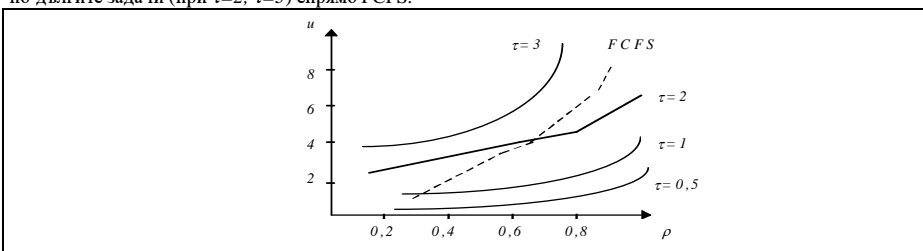
SJF (Shortest Job First) - за обработка с приоритет се планират по-късите задачи, които имат по-малка трудоемкост (минимизация на средното време за отговор u).

Сравнение на методите FCFS и SJF може да се направи на базата на $u(\rho)$ - средно време за отговор, където $\rho = \lambda \Theta$ е относителното натоварване на КС, като се предполага, че входният поток от заявки е с равномерна интензивност λ и експоненциален закон на разпределение за времената на обработка τ_i със средна стойност Θ . Могат да се направят следните изводи:

а) SJF води до намаляване на u за по-късите задачи ($\tau=0.5$) за сметка на по-дългите ($\tau=3$). Времето за престой (отговор) u при средните задачи ($\tau=1$) е по-малко от това при FCFS;

б) FCFS има по-стръмна зависимост $u(\rho)$, т.е. при нарастване на натоварването (заявки във входната опашка) силно се забавя обработката в КС;

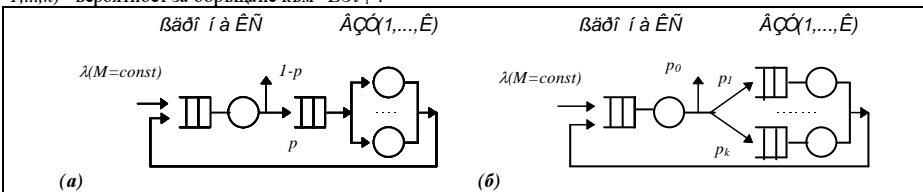
в) При фиксирано натоварване ρ (напр. $\rho=0.4$) методът SJF дава по-голямо времето за отговор u за по-дългите задачи (при $\tau=2, \tau=3$) спрямо FCFS.



♦ *Изследване на мултипрограмен КА с пакетна обработка.* Архитектурната организация предполага разделяне на основната обработка (ядро на КС) от допълнителните В/И операции (периферия) и наличие на двуслойна памет - оперативна (ОП) и външна (ВП). Ядро на КС (процесор и ОП) има средно време за процесорно обслужване Θ_1 . ВП е организирана от K ВЗУ със средно време за В/И обработка Θ_2 . Представянето на КА може да се извърши чрез СМО, като са възможни следните абстрактни модели:

а) *КС с обща опашка към външната памет* - вероятността за обръщане на една задача към ВП е ρ , като обслужването във В/И канали е по метода FCFS (FIFO) с управление от един контролер;

б) *КС със самостоятелни опашки към външната памет* - има K канални процесора (селекторни канали) и вероятности: p_0 - вероятност за завършване на текущата фаза от обработката на процесора; p_i ($i=1, \dots, k$) - вероятност за обръщане към "ВЗУ i ".



Обект на изследване е относителното натоварване на процесора ρ като функция от Θ_2/Θ_1 при фиксирани брой канали ($K=const$) и брой задачи (коэффициент на мултипрограмиране $M=const$). Примерно изследване позволява следните обобщени изводи:

- а) нарастването на M ($K=const$) води до увеличено натоварване на процесора;
- б) при $K=1$ и малки стойности за Θ_2/Θ_1 , ρ расте силно с нарастване на M ;
- в) при $K>1$ с $M \rightarrow \infty$ значително се разширява областта от пълно натоварване на процесора поради доброто съвместяване на различните обработки;

г) при $M > 1$, $K > 2$ натоварването ρ за КА със самостоятелни опашки е по-малко в сравнение с КА с обща опашка.

♦ Оптимален коефициент на мултипрограмиране. При определен състав на КС нарастване на производителността се получава чрез увеличаване на M . Това, обаче, води до увеличаване на ОП, което оскъпява КС. Под оптимален коефициент на мултипрограмиране M_{opt} се разбира такава стойност, при която функцията на цената на КС $C(M) = (S' + S_0 \cdot M) / \pi_0(M)$ е минимална, където: S' - цена на непроменящата се техническа част; $S_0 \cdot M$ - цената на разширената M пъти ОП; $\pi_0(M)$ - общата производителност при коефициент на мултипрограмиране M .

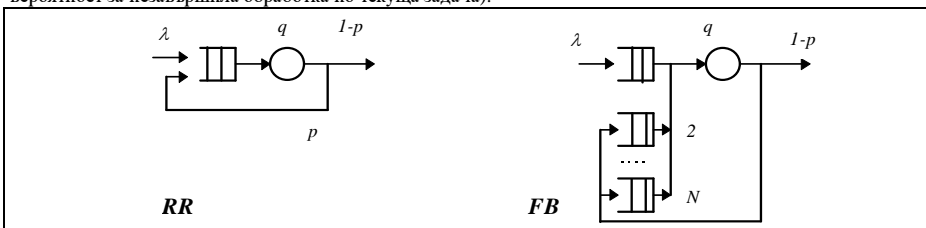
1.4. Особенности на диспечерирането при мултипрограми КС с времеделене

♦ Базови методи за диспечериране RR и FB. Предвиждат циклично изпълнение на задачите при неизвестни предварително трудоемкости τ_i . Състоят се в следното:

RR (Round Robin) - режим на циклична обработка на задачите с фиксиран квант време q . Ако обработката не е завършила, задачата се подрежда във входната опашка за следваща обработка. Характерна е за КС с телеобработка, времеделене и при мрежови архитектури.

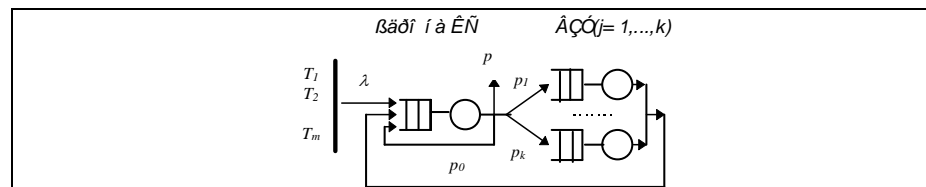
FB (Foreground-Background) - реализира алгоритъм RR при няколко нива (опашки) за приоритетно обслужване. Ако задача Z_i не е завършила през първия квант, тя се изпраща в по-долна опашка (следващо ниво). При всяко незавършване задачата "пропада" с едно ниво надолу. Ако броят на необходимите кванти надхвърля броя на нивата N , задачата зацикля в най-долната опашка. Към обработка на задача от ниво $(j+1)$ се преминава само ако са изчерпани всички задачи от ниво (j) . Допуска се постоянен или променлив квант q .

Обобщени модели на двете стратегии (λ - интензивност на входен поток от задачи за обслужване; p - вероятност за незавършила обработка по текуща задача):



♦ Изследване на мултипрограми КС с времеделене. Базови режими при КС с диспечериране RR и FB са времеделене и телеобработка. Основната цел е минимизиране на средното време за отговор u на задачите, като не се допуска образуване на големи опашки пред устройствата. Всички задачи в КС са активни независимо от техния брой и се изпълняват в режим на мултипрограмиране с променлив коефициент M . Входният поток от заявки е неограничен, натоварването на КС при обслужване на терминалната мрежа е сравнително малко, има няколко селекторни канали за връзка на КС-ядрото с ВЗУ.

По-долу е показано примерно представяне чрез СМО на КА с времеделене със следните параметри: интензивност на входния поток λ ; средно време за процесорна обработка θ_j ; средно време за обмен с ВЗУ θ_j ($j=1,2,\dots,k$); вероятност за обръщане към устройство от КС p_i ($i=1,\dots,k$); вероятност за напускане на системата от обслужена задача $p = 1 - (p_0 + p_1 + \dots + p_k)$.



След провеждане на примерно изследване могат да се направят изводите:

а) с нарастване на средния коефициент на мултипрограмиране M нараства и средното натоварване на процесора ρ , като се достига до насищане на зависимостта;

б) при различни фиксирани стойности за отношението $\gamma = \theta_j / \theta_{vj}$ (θ_{vj} е средно време за достъп до ВЗУ) се получава фамилия от зависимости $\rho(M)$; $\gamma = const$, за които нарастването на γ води до значително нарастване на ρ при фиксирани стойности за M ;

в) за фиксирана стойност на ρ минималната стойност за M е при най-ниска стойност за γ , т.е. колкото по-малко е времето за обръщане към ВЗУ, толкова по-ниска е стойността за M .

♦ Оптимален коефициент на балансираност. При КС с времеделене с увеличаване на бърздействието V_i на отделните ресурси ($i=1,\dots,k$) и при еднакви други условия, времето за отговор u на системата намалява. Увеличаването на бърздействието е добре за потребителите, но изисква допълнителни разходи. Целта е да се намери оптимално бърздействие V_{opt} на ресурсите, за което сумарните загуби в системата да са минимални, т.е. $C(V) = C1(V) + C2(V) = min$, където: $C1(V)$ - загуби от престой на ресурсите; $C2(V)$ - загуби от забавяне на обработката. Тогава V_{opt} е стойността на V , за която $C(V_{opt}) = min$

2. Задание за лабораторна работа

1. Да се разучат основните режими на работа и особеностите при диспечериране на независими процеси в КС.

2. Да се изследва мултипрограмна SISD архитектура с единичен дисков тракт и метод на диспечериране FCFS за равномерно разпределения входен поток от задачи. Да се симулира работата с вариране на интензивността на входния поток като се анализират моделните резултати и се построят функционални зависимости за базови системни характеристики.

3. Да се модифицира симулационния модел от задача (2) съгласно представените в т.1.3 два абстрактни модела на КС с външна памет и постоянен коефициент на мултипрограмиране. Да се проведат симулационни експерименти и резултатите да се сравнят с предходните.

4. Да се изследва мултипрограмна КС с алгоритъм за планиране SJF. Да се проведе симулационно изследване и се анализират получените резултати.

5. Да се изследва стратегия за планиране RR чрез симулационни експерименти с вариране на интензивността λ на входния поток (при $q = const$) и с вариране на кванта q (при $\lambda = const$). Да се построят графично зависимости $\rho(\lambda)$; $q = const$ и $\rho(q)$; $\lambda = const$ за натоварването на ЦП. Аналогично изследване да се проведе за представената в т.1.4 мултипрограмна КА с времеделене.

6. Да се проведе вариационно симулационно изследване на стратегия за планиране FB при $N=3$ нива за разпределяне на задачите. Аналогично на предходната задача да се построят графично зависимости $\rho(\lambda)$; $q = const$ и $\rho(q)$; $\lambda = const$ за натоварването на ЦП.

3. Лабораторни експерименти

По точка (1)

Запознаване с текстовия материал от теоретичната постановка.

По точка (2)

За изследване на указаната мултипрограмна SISD архитектура се съставя *абстрактно описание на базата на СМО*, което е подобно на тези от т.1.3 при $K=1$. Задачите постъпват на входа през интервали ($1/\lambda$) равномерно разпределени в диапазона [5,15] ЕМВ. След обслужване в ЦП вероятността задачата да е завършена е 60%. Останалата част от задачи изискват достъп до външната памет и допълнителна процесорна обработка. Симулацията се изпълнява чрез *програмен модел* MODEL3 от средата, като резултатите се обобщават в таблица 1. Оценките се анализират и се построяват графично функционалните зависимости $\rho(\lambda)$ и $\Theta_i(\lambda)$.

Таблица 1.

Интензивност на вх. поток	Средно натоварване ρ		Средно време за задача Θ_i	
	CPU	DISK	CPU	DISK
0,01 = 1/100				
0,02 = 1/50				
0,05 = 1/20				
0,10 = 1/10				
0,20 = 1/5				

По точка (3)

В предходния модел се въвеждат необходимите блокове за описание на допълнителните архитектурни компоненти (дискони трактове) съгласно абстрактните описания на КА от т.1.3. За представяне на $M=const$ (напр. $M=10$) се използва блок: *GENERATE 0,0,0,10*. Новият модел се съхранява под друго име. Провеждат се експерименти аналогично на задача (2).

По точка (4)

За изследване на мултипрограмна КС с алгоритъм за планиране SJF се използват абстрактното представяне и симулационният модел, представени по-долу. Предвидена е обработка на пакет от 40 задачи (виж *START 40*) от три типа - къси, средни, дълги. Крайните оценки, снети от "Стандартна статистика", се записват в таблица 2.

Таблица 2.

Име	Средно натоварване	Средно съдържание	Средно време за престой
CPU			
QPR1	-		
QPR2	-		
QPR3	-		

По точки (5) и (6)

Симулацията се извършва на базата на обобщените модели RR и FB от т.1.4 при $q=const$. Симулационните експерименти се провеждат при задаване на конкретни стойности за интензивността λ (чрез поле A на блок GENERATE), вероятността p и кванта q . Изследва се средното натоварване ρ на процесора при двете стратегии за зависимостите:

- а) $\rho = \rho(\lambda; q=4)$ - за $\lambda = 0,5; 0,1; 0,15; 0,2;$
- б) $\rho(q; \lambda=1/A=0,2)$ - за $q = 2; 4; 6; 8.$

Резултатите се оформят в таблици и се построяват графично зависимостите $\rho(\lambda)$ и $\rho(q)$.

Абстрактен модел за SJF:	Програмен модел RR:
	generate A,B -- A=?; B<A
	queue qcpu
	seize cpu

	<pre>departqcpu advance q -- q=? release cpu transfer p,,back -- p∈[0,1]=? terminate generate 500 terminate 1 start 1 end</pre>
<p><u>Програмен модел SJF:</u></p> <pre>generate 60,30,,,10 queue qpr1 transfer ,lab0 generate 80,20,,,30 queue qpr2 transfer ,lab0 generate 50,20,,,50 queue qpr3 lab0 test e q\$qpr1,0,lab1 lab0 test e q\$qpr2,0,lab2 lab0 test e q\$qpr3,0,lab3 lab1 transfer ,lab0 lab1 seize cpu lab1 depart qpr1 lab2 transfer ,next lab2 seize cpu lab2 depart qpr2 lab3 transfer ,next lab3 seize cpu lab3 depart qpr3 next advance pr,8 next release cpu next terminate 1 next start 40 next end</pre>	<p><u>Програмен модел FB:</u></p> <pre>generate A,B -- A=?; B<A queue 1 lab1 test ne q1,0,lab2 lab1 seize cpu lab1 depart 1 lab1 advance q -- q=? lab1 release cpu lab1 transfer ,.5,,out lab1 queue 2 lab1 transfer ,lab1 lab2 test ne q2,0,lab3 lab2 seize cpu lab2 depart 2 lab2 advance q -- q=? lab2 release cpu lab2 transfer ,.5,,out back queue 3 back transfer ,lab1 lab3 test ne q3,0,out lab3 seize cpu lab3 depart 3 lab3 advance q -- q=? lab3 release cpu lab3 transfer ,.5,,back out terminate out generate 500 out terminate 1 out start 1 out end</pre>

4. Съдържание на отчета

В отчета, оформен стандартно (име, факултетен номер, група, курс, дисциплина), се включва:

- а) номер и тема на упражнението;
- б) задание за лабораторна работа;
- в) резултати от лабораторните експерименти 2, 3, 4, 5 и 6 в таблица 1.