Kumpulan Soal Ujian Sistem Operasi 2010-2015

Rahmat M. Samik-Ibrahim et. al.

http://rms46.vLSM.org/2/183.pdf

Berikut merupakan soal ujian yang pernah diberikan di Fakultas Ilmu Komputer, Universitas Indonesia (Fasilkom UI) antara tahun 2010 dan seterusnya. Kumpulan ini merupakan kontribusi bersama dari Rahmat M. Samik-Ibrahim (VauLSMorg), Muhammad H. Hilman (Fasilkom UI), Heri Kurniawan (Fasilkom UI), Amril Syalim (Fasilkom UI), Muhammad Anwar Ma'sum (Fasilkom UI), et.al.

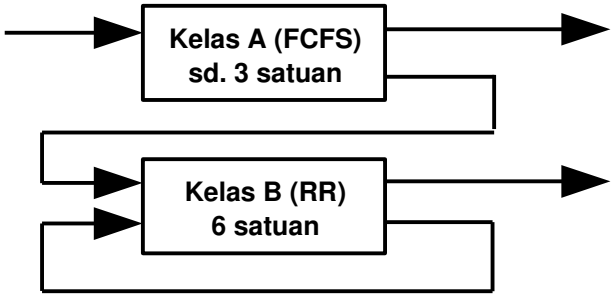
Table of Contents

Istilah Yang Digunakan

ACL: Access Control List
C/H/S: Cylinder/Head/Sector
COW: Copy On Write
FAT: File Allocation Table
HDD: Hard Disk Drive
MMU: Memory Management Unit
PA: Physical Address
PF: Physical Frame
PFN: Physical Frame Number
PM: Physical Memory
PTBR: Page Table Base Register
PTE: Page Table Entry
RAID: Redundant Array of Independent Disks
SJF: Shortest Job First
SSF: Shortest Seek First
TLB: Translation Look-aside Buffer
VA: Virtual Address
VFS: Virtual File System
VM: Virtual Memory
VP: Virtual Page
VPN: Virtual Page Number

(2010-1) Penjadwalan Proses Bertingkat

Sebuah sistem *preemptive* yang terdiri dari dua kelas penjadwal bertingkat: kelas A dan kelas B. Kedua penjadwal tersebut berfungsi secara bergiliran dengan perbandingan 4:1 (4 *burst* kelas A, lalu 1 *burst* kelas B). Setiap CPU *burst* baru akan diekskusi secara **FCFS** (*First Come First Served*) oleh penjadwal kelas A. Burst tidak rampung dalam 3 (tiga) satuan waktu, akan dialihkan ke penjadwal kelas B yang berbasis **RR** (*Round Robin*) dengan kuantum 6 (enam) satuan waktu. Abaikan "waktu alih" (*switching time*).



Diketahui P1(0: 13), P2(2: 1), P3(4: 5), P4(6: 1), P5 (8: 5) dimana **Px(Y: Z)** berarti: "**burst Proses X, mulai saat Y selama Z satuan waktu**". Gunakan notasi sebagai berikut:

A(k): Penjadwal kelas A, sisa burst = k satuan.
B(m): Penjadwal kelas B, sisa burst = m satuan.
W(n): Waktu tunggu = n satuan.

Lengkapi tabel berikut ini:

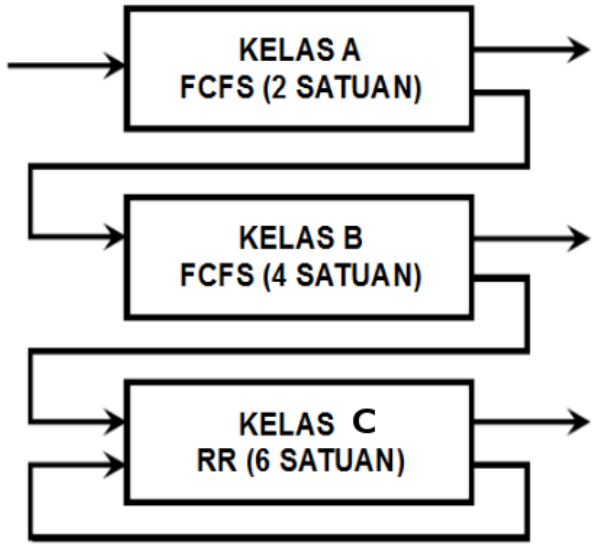
	0	1	2	3	4	5	6	7	8	9	10	11	12
P1	A(13)	A(12)	A(11)	W(1)	W(2)	W(3)	W(4)	W(5)	B(10)	B(9)	B(8)	B(7)	B(6)
P2			W(1)	A(1)	-	-	-	-	-	-	-	-	-
P3					A(5)	A(4)	A(3)	W(1)	W(2)	W(3)	W(4)	W(5)	W(6)
P4							W(1)	A(1)	-	-	-	-	-
P5									W(1)	W(2)	W(3)	W(4)	W(5)

	13	14	15	16	17	18	19	20	21	22	23	24	25
P1													
P2													
P3													
P4													
P5													

Berapa waktu tunggu (W) dari masing-masing proses?

W(P1) = _____; W(P2) = _____; W(P3) = _____; W(P4) = _____; W(P5) = _____

(2013-1) Penjadwalan Proses Bertingkat



Sebuah sistem *preemptive* terdiri dari tiga kelas penjadwal bertingkat: kelas A, B, dan C. Ketiga penjadwal tersebut berfungsi secara bergiliran dengan perbandingan 4:2:1 (4 *burst* kelas A, lalu 2 *burst* kelas B, lalu 1 *burst* kelas C).

Setiap CPU burst baru akan dieksekusi oleh penjadwal kelas A yang berbasis *First Come First Served* (**FCFS**) dengan dengan batasan 2 satuan waktu. Apabila burst tidak rampung dalam sekali jalan, proses akan dialihkan ke penjadwal kelas B yang berbasis FCFS dengan batasan 4 satuan waktu.

Demikian pun apabila burst tidak rampung, proses akan dialihkan ke penjadwal kelas C yang berbasis *Round Robin* (**RR**) dengan kuantum 6 satuan waktu. Abaikan waktu alih (*switching time*).

Diketahui P1(0:13), P2(2:2), P3(4:5), P4(6:1), P5(8:5) dimana **Px(Y:Z) berarti: "burst proses X, mulai saat Y selama Z satuan waktu"**. Gunakan notasi sebagai berikut:
A(k): Penjadwal kelas A, sisa burst = k satuan.
B(k): Penjadwal kelas B, sisa burst = k satuan.
C(k): Penjadwal kelas C, sisa burst = k satuan.

a. Lengkapi tabel berikut

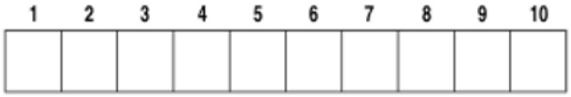
	0	1	2	3	4	5	6	7	8	9	10	11	12
P1	A(13)	A(12)	W(1)	W(2)	W(3)								
P2			A(2)	A(1)	-								
P3					A(5)								
P4													
P5													

	13	14	15	16	17	18	19	20	21	22	23	24	25
P1													
P2													
P3													
P4													
P5													

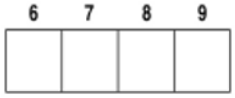
b. Hitung "*waiting time*" (**W**) dari masing-masing proses.

(2012-2) Penjadwalan Thread

a) Tuliskan NPM anda (10 digit)



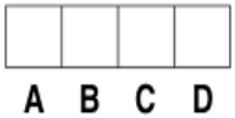
b) Salin digit ke 6 hingga ke 9 (4 digit)



c) Sortir digit-digit tersebut dengan bobot:

0 > 9 > 8 > 7 > 6 > 5 > 4 > 3 > 2 > 1

petakan ke variabel A, B, C, dan D



Contoh: [1106436021] → [3602] → [0632]

A = 0 B = 6 C = 3 D = 2

Diketahui, fungsi "**`day_month(X)`**" dengan pemetaan sebagai berikut:

X	1	2	3	4	5	6	7	8	9	0
day_month(X)	0	31	59	90	120	151	181	212	243	273

Diketahui, fungsi "**`init_n_start_thread(T1,T2,....Tn)`**". Pada akhir pengeksekusian fungsi tersebut, akan terbentuk "**n**" buah ***thread non-preemptive*** T1, T2, … Tn. Setiap ***thread*** akan dimasukkan ke dalam antrian "***Ready Queue***" dengan status "**Rd**" (Ready). Apabila ada CPU yang kosong, status ***thread*** kepala antrian akan berubah menjadi "**R**" (Run). Fungsi "**`init_n_start_thread()`**" hanya dapat dieksekusi ulang, jika **SEMUA** thread terdahulu telah rampung.

"***Pseudo-code***" pada halaman berikut memerlukan satuan waktu eksekusi sebagai berikut:

No	*Jenis*	*Satuan*	*Contoh Baris*
1	Awal M (fungsi Main/Utama)	1 (R)	`01 M: {`
2	Awal T (Thread)	1 (R)	`05 T1:{`
3	Eksekusi Baris	1 (R) / baris	`16 day1 = dm1 + D;`
4	Eksekusi Fungsi	2 (RR) / baris	`10 dm1 = day_month(B);` `20 init_n_start_thread(T9);`

d) Hitunglah, **delta = _____**

```
##  Program berikut terdiri dari M: (main) dan T1...T9 (threads).
##  Nilai A, B, C, D sesuai dengan NPM. Lihat halaman sebelumnya!

01 M: {
02       A = _________ ;
03       B = _________ ;
04       init_n_start_thread(T1, T2, T3, T4, T5);
     }
##
05 T1:{
06       C = _________ ;
     }
##
07 T2:{
08       D = _________ ;
     }
##
09 T3:{
10       dm1 = day_month(B);
     }

##
11 T4:{
12       dm2 = day_month(A);
     }

##
13 T5:{
14       init_n_start_thread(T6, T7, T8);
     }

##
15 T6:{
16       day1 = dm1 + D;
     }

##
17 T7:{
18       day2 = dm2 + C;
     }

##
19 T8:{
20       init_n_start_thread(T9);
     }

##
21 T9:{
22       delta = day2 - day1;
     }
```

Lengkapi diagram berikut ini untuk SATU PROSESOR (P1). Lihat contoh.													
T	Baris-P1	P1	M	T1	T2	T3	T4	T5	T6	T7	T8	T9	
1	01	M	R	-	-	-	-	-	-	-	-	-	
2	02	M	R	-	-	-	-	-	-	-	-	-	
3	03	M	R	-	-	-	-	-	-	-	-	-	
4	04	M	R	-	-	-	-	-	-	-	-	-	
5	04	M	R	-	-	-	-	-	-	-	-	-	
6	05	T1	-	R	Rd	Rd	Rd	Rd	-	-	-	-	
7	06	T1	-	R	Rd	Rd	Rd	Rd	-	-	-	-	
8	07	T2	-	-	R	Rd	Rd	Rd	-	-	-	-	
9	08	T2	-	-	R	Rd	Rd	Rd	-	-	-	-	
10	09	T3	-	-	-	R	Rd	Rd	-	-	-	-	
11													
12													
13													
14													
15													
16													
17													
18													
19													
20													
21													
22													
23													
24													
25													
26													
27													
28													
29													
30													

Revision: 716 -- 15 Nov 2017 URL: http://rms46.vLSM.org/2/183.pdf

Lengkapi diagram berikut ini untuk DUA PROSESOR (P1+P2). Lihat contoh.

T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Baris-1	01	02	03	04	04	05	06	09	10	10																				
P1	M	M	M	M	M	T1	T1	T3	T3	T3																				
Baris-2	-	-	-	-	-	07	08	11	12	12																				
P2	-	-	-	-	-	T2	T2	T4	T4	T4																				
M	R	R	R	R	R	-	-	-	-	-																				
T1	-	-	-	-	-	R	R	-	-	-																				
T2	-	-	-	-	-	R	R	-	-	-																				
T3	-	-	-	-	-	Rd	Rd	R	R	R																				
T4	-	-	-	-	-	Rd	Rd	R	R	R																				
T5	-	-	-	-	-	Rd	Rd	Rd	Rd	Rd																				
T6	-	-	-	-	-	-	-	-	-	-																				
T7	-	-	-	-	-	-	-	-	-	-																				
T8	-	-	-	-	-	-	-	-	-	-																				
T9	-	-	-	-	-	-	-	-	-	-																				

Revision: 716 -- 15 Nov 2017 URL: http://rms46.vLSM.org/2/183.pdf

Lengkapi diagram berikut ini untuk LIMA PROSESOR (P1, P2,... P5). TANPA contoh.

T	Baris-1	P1	Baris-2	P2	Baris-3	P3	Baris-4	P4	Baris-5	P5	M	T1	T2	T3	T4	T5	T6	T7	T8	T9
1																				
2																				
3																				
4																				
5																				
6																				
7																				
8																				
9																				
10																				
11																				
12																				
13																				
14																				
15																				
16																				
17																				
18																				
19																				
20																				
21																				
22																				
23																				
24																				
25																				
26																				
27																				
28																				
29																				
30																				

Revision: 716 -- 15 Nov 2017 URL: http://rms46.vLSM.org/2/183.pdf

(2014-1) Penjadwalan

a) Untuk penjadwal ideal, bagaimanakah sebaiknya nilai (max atan min) dari parameter-parameter berikut: "*response time*", "*turnaround time*", "*throughput*"?

b) Dari ketiga parameter tersebut di atas, manakah yang paling penting untuk sebuah server? Terangkan!

c) Dari ketiga parameter tersebut di atas, manakah yang paling penting untuk sebuah proses interaktif? Terangkan!

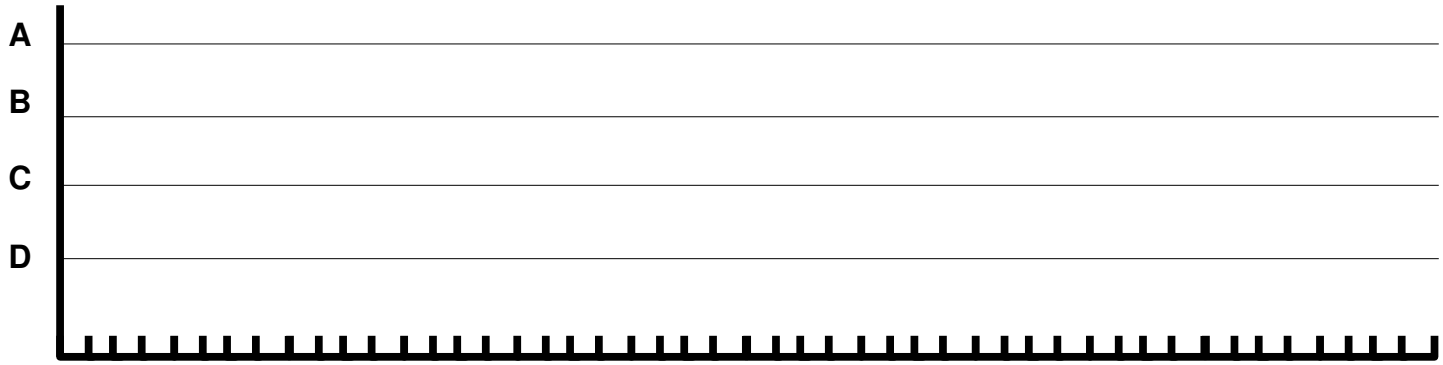
d) Terangkan masalah apa pada sebuah proses interaktif yang ingin diperbaiki/ditanggulangi oleh CFS?

(2010-1) Status Proses

Diketahui empat proses, A(90: 17.2), B(80: 24.5), C(70: 10.5), D(60: 30); **[**W(X: Y); W=nama proses; X= I/O Wait(%); Y=waktu CPU**]** mulai saat bersamaan, dengan tabel utilitas CPU dan tabel kombinasi derajat multi-program sebagai berikut:

	Kombinasi Multiprogram (%)														
	A	B	C	D	A+B	A+C	A+D	B+C	B+D	C+D	A+B+C	A+B+D	A+C+D	B+C+D	A+B+C+D
Utilitas CPU per proses A	10	-	-	-	9.3	9.3	9.2	-	-	-	8.3	8.1	7.8	-	7
Utilitas CPU per proses B	-	20	-	-	19	-	-	18	17	-	17	16	-	15	14
Utilitas CPU per proses C	-	-	30	-	-	28	-	26	-	25	25	-	23	22	21
Utilitas CPU per proses D	-	-	-	40	-	-	37	-	35	33	-	32	31	30	28

Gambar relasi antara proses dan waktu sebagai berikut:



Berapakah waktu total program D, jika sepenuhnya berjalan sendirian?

(2011-2) Status Proses

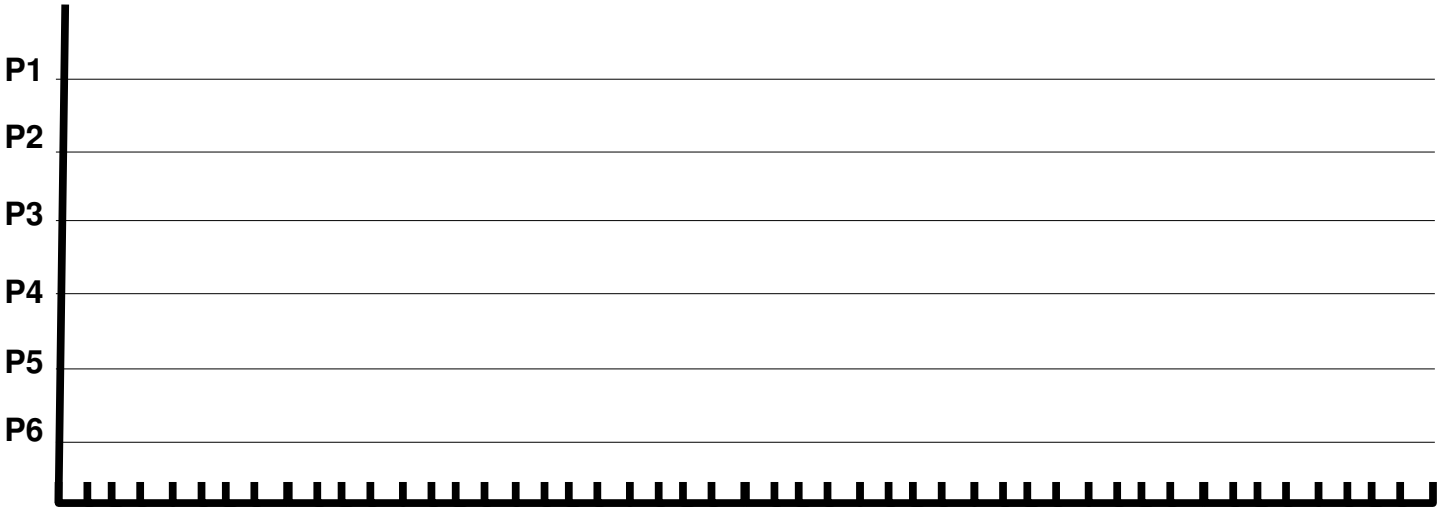
Diketahui enam (6) proses homogen (sejenis) yang menggunakan (waktu) CPU masing-masing 78 detik. Jika hanya satu proses berjalan (derajat multiprogram=1), maka perbandingan utilisasi waktu CPU ialah 10%. Untuk derajat multiprogram 2 -- 3 -- 4 -- 5 -- 6, maka perbandingan utilisasi waktu CPU berturut-turut 9.5% -- 9% -- 8,6% -- 8,2% -- 7,8%.
a) Hitung, berapa waktu total yang diperlukan untuk menjalankan secara bersamaan (concurrently) ke-enam proses tersebut.
b) Hitung, berapa waktu total yang diperlukan, untuk menjalankan secara berkesinambungan (satu per satu), ke-enam proses tersebut.

(2011-1) Status Proses

Diketahui enam proses, P1 (B, 0, 14.8), P2 (B, 0, 45), P3 (B, 0, 57.8), P4 (B, 0, 77.8), P5 (A, 200, 53.2) dan P6 (A, 300, 25.4); **[**(X, Y, Z)); X= Jenis proses (A atau B), Y=waktu start, Z=waktu CPU**]** dengan tabel utilitas CPU(%) dan tabel kombinasi derajat multi-program sebagai berikut:

	A	B	AA	AB	BB	AAA	AAB	ABB	BBB	AAAA	AAAB	AABB	ABBB	BBBB
A	40	-	32	34.6	-	26.1	28.6	30.8	-	21.8	23.6	25.4	27.8	-
B	-	20	-	17.3	18	-	14.3	15.4	16.3	-	11.8	12.8	13.9	14.8

Gambar relasi antara proses dan waktu sebagai berikut:



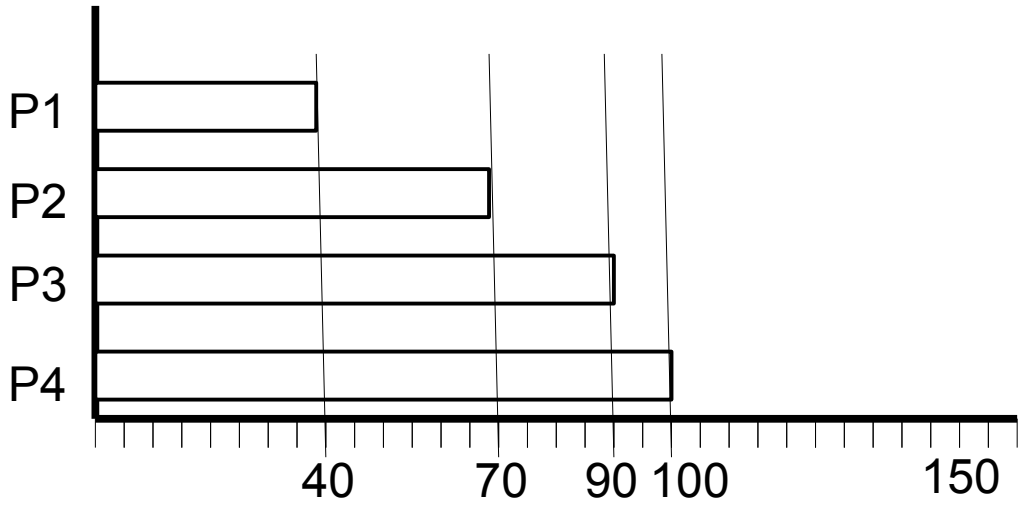
Berapa waktu total yang dibutuhkan P1, jika berjalan sendiri tanpa proses-proses yang lain?

(2012-1) Status Proses

Diketahui empat proses, P1(A), P2(A), P3(B), P4(B), yang secara **paralel** mulai dari t = 0. "A" dan "B" merupakan jenis proses dengan tabel utilitas CPU(%) dan tabel kombinasi derajat multi-program sebagai berikut:

	A	B	AA	AB	BB	AAA	AAB	ABB	BBB	AAAA	AAAB	AABB	ABBB	BBBB
A	40	-	32	34.6	-	26.1	28.6	30.8	-	21.8	23.6	25.4	27.8	-
B	-	20	-	17.3	18	-	14.3	15.4	16.3	-	11.8	12.7	13.9	14.8

"**Waktu total**" proses berturut-turut, P1=40, P2=70, P3=90, dan P4=100. Relasi antara proses dan waktu sebagai berikut:



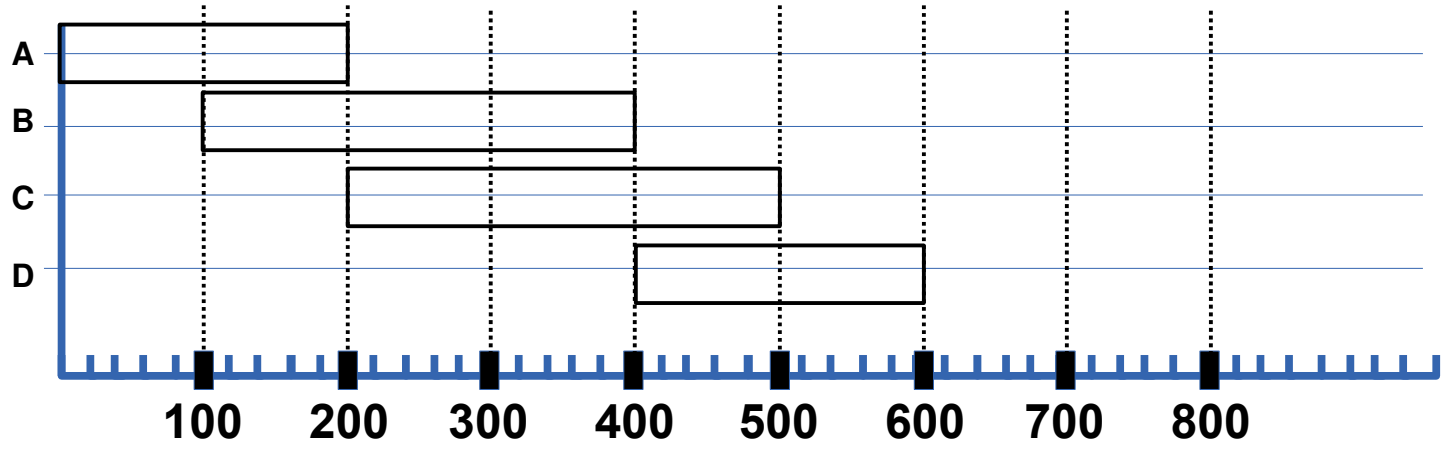
a) Hitung ***CPU time*** dari P1!
b) Hitung ***CPU time*** dari P2!
c) Hitung ***CPU time*** dari P3!
d) Hitung ***CPU time*** dari P4!
e) Hitung waktu total yang diperlukan, jika P1, P2, P3, dan P4 dijalankan secara **serial**/bergantian! (Jika terjadi angka pecahan; **TIDAK USAH** dihitung lengkap).

(2013-2) Status Proses

Diketahui empat proses, A(90%), B(80%), C(70%), D(60%); **[**W(X); W=nama proses; X= I/O Wait(%)**]** dengan tabel utilitas CPU dan tabel kombinasi derajat multi-program sebagai berikut:

	Kombinasi Multiprogram (%)														
	A	B	C	D	A+B	A+C	A+D	B+C	B+D	C+D	A+B+C	A+B+D	A+C+D	B+C+D	A+B+C+D
Utilitas CPU per proses A	10	-	-	-	9.3	9.3	9.2	-	-	-	8.3	8.1	7.8	-	7
Utilitas CPU per proses B	-	20	-	-	19	-	-	18	17	-	17	16	-	15	14
Utilitas CPU per proses C	-	-	30	-	-	28	-	26	-	25	25	-	23	22	21
Utilitas CPU per proses D	-	-	-	40	-	-	37	-	35	33	-	32	31	30	28

Gambar relasi antara proses dan waktu sebagai berikut:



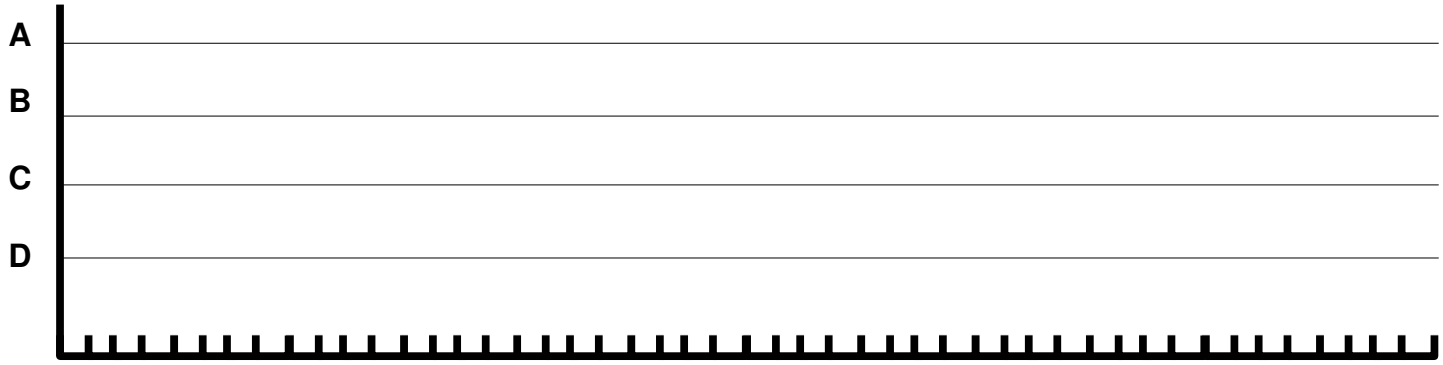
a) Hitung ***CPU time*** untuk proses A.
b) Hitung ***CPU time*** untuk proses B.
c) Hitung ***CPU time*** untuk proses C.
d) Hitung ***CPU time*** untuk proses D.
e) Berapakah waktu total proses D, jika sepenuhnya berjalan sendirian?

(2014-2) Status Proses

Diketahui empat proses; A(18.6), B(19), C(?), D(33); **[**W(X); W=nama proses; X=waktu CPU]. Proses A dan B mulai pada saat yang bersamaan. Proses D mulai saat proses A selesai. Proses C mulai saat proses B selesai. Proses C dan D selesai bersamaan. Berikut tabel utilitas CPU dan tabel kombinasi derajat multi-program.

	Kombinasi Multiprogram (%)														
	A	**B**	**C**	**D**	**A+B**	**A+C**	**A+D**	**B+C**	**B+D**	**C+D**	**A+B+C**	**A+B+D**	**A+C+D**	**B+C+D**	**A+B+C+D**
Utilitas CPU per proses A	10	-	-	-	9.3	9.3	9.2	-	-	-	8.3	8.1	7.8	-	7
Utilitas CPU per proses B	-	20	-	-	19	-	-	18	17	-	17	16	-	15	14
Utilitas CPU per proses C	-	-	30	-	-	28	-	26	-	25	25	-	23	22	21
Utilitas CPU per proses D	-	-	-	40	-	-	37	-	35	33	-	32	31	30	28

a) Lengkapi diagram berikut:



b) Hitung waktu CPU dari proses C.

(2015-1) Status Proses

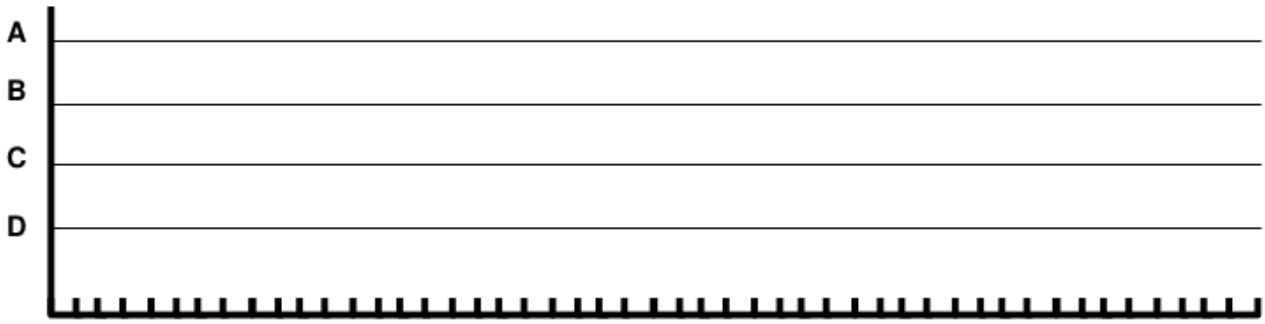
	Kombinasi Multiprogram (%)														
	A	B	C	D	A+B	A+C	A+D	B+C	B+D	C+D	A+B+C	A+B+D	A+C+D	B+C+D	A+B+C+D
Utilitas CPU per proses A	10	-	-	-	9.3	9.3	9.2	-	-	-	8.3	8.1	7.8	-	7
Utilitas CPU per proses B	-	20	-	-	19	-	-	18	17	-	17	16	-	15	14
Utilitas CPU per proses C	-	-	30	-	-	28	-	26	-	25	25	-	23	22	21
Utilitas CPU per proses D	-	-	-	40	-	-	37	-	35	33	-	32	31	30	28

Proses “D” dijalankan selama 1100 satuan waktu sejak t=0.
Proses “A” hanya dijalankan, selama proses “D” berjalan.
Proses “B” hanya dijalankan, jika waktu CPU (**CPU TIM**E) proses “D” lebih dari 310 satuan waktu.
Proses “C” hanya dijalankan, selama proses “B” tidak berjalan.

a) Sebutkan, proses mana saja yang akan dijalankan pada saat t = 0.

b) Periksa, apakah proses “B” akan dijalankan. Jika “TIDAK”, mengapa? Jika “YA”, mulai kapan?

c) Lengkapi diagram berikut ini:



d) Berapa waktu total CPU (CPU TIME) dari proses “A”

e) Berapa waktu total CPU (CPU TIME) dari proses “C”

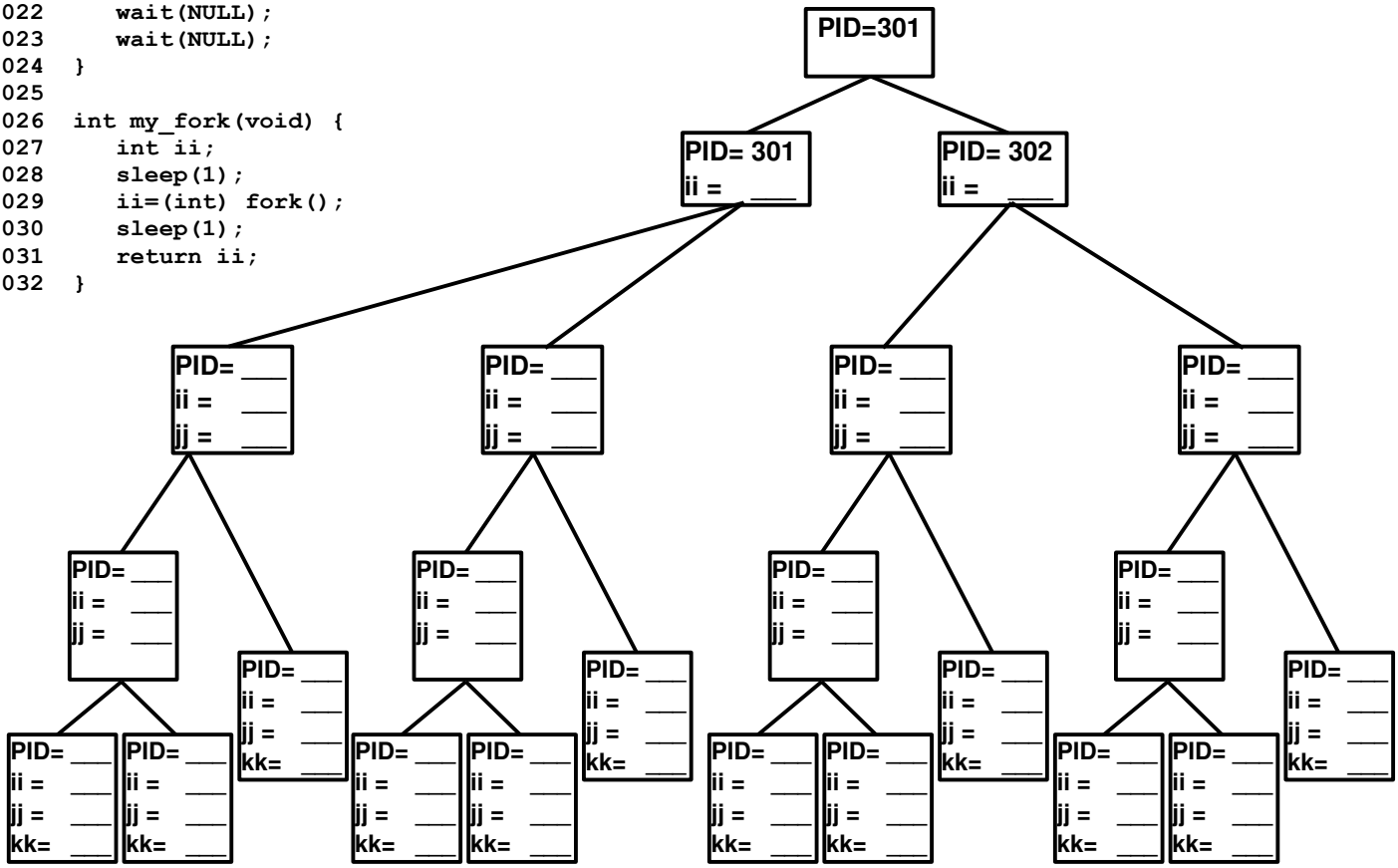
f) Berapa waktu total CPU (CPU TIME) dari proses “D”

(2010-1) Fork

Lengkapi kotak serta pohon (*tree*) hasil kompilasi program "fork2010.c" (PID=301) berikut ini:

```
001  #include <sys/types.h>
002  #include <sys/wait.h>
003  #include <stdio.h>
004  #include <unistd.h>
005
006  int my_fork(void);
007
008  main(void)
009  {
010     int ii, jj, kk;
011
012     my_fork();
013     ii = (int) getpid();
014     my_fork();
015     jj = (int) getpid();
016     if (my_fork() >  0)
017        my_fork();
018     kk = (int) getpid();
019     printf ("ii = %3.3d -- jj = %3.3d -- kk = %3.3d\n",ii,jj,kk);
020     wait(NULL);
021     wait(NULL);
022     wait(NULL);
023     wait(NULL);
024  }
025
026  int my_fork(void) {
027     int ii;
028     sleep(1);
029     ii=(int) fork();
030     sleep(1);
031     return ii;
032  }
```

```
ii = ______ -- jj = 301 -- kk = 301
ii = ______ -- jj = 302 -- kk = 302
ii = 302    -- jj = 303 -- kk = 303
ii = ______ -- jj = 304 -- kk = 304
ii = ______ -- jj = ______ -- kk = 305
ii = ______ -- jj = 303 -- kk = 306
ii = ______ -- jj = 302 -- kk = 307
ii = ______ -- jj = 301 -- kk = 308
ii = ______ -- jj = 301 -- kk = 309
ii = ______ -- jj = 304 -- kk = 310
ii = ______ -- jj = 303 -- kk = 311
ii = ______ -- jj = 302 -- kk = 312
```



Berapakah kisaran dari ii, jj, dan kk?

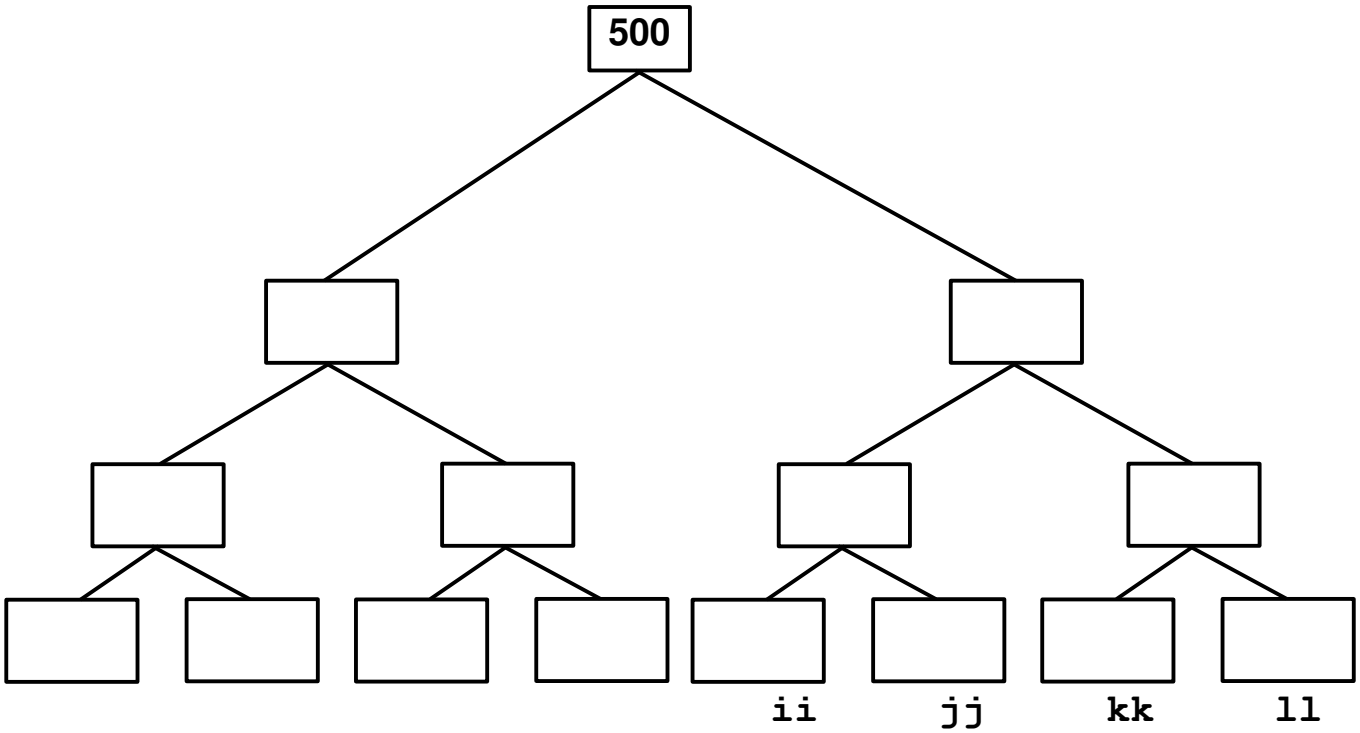
☐ ≤ ii ≤ ☐ ☐ ≤ jj ≤ ☐ ☐ ≤ kk ≤ ☐ ii ☐ jj ☐ kk

Apa guna “wait(NULL)” pada baris 20-23 di atas?

Revision: 716 -- 15 Nov 2017 URL: http://rms46.vLSM.org/2/183.pdf

(2011-1) Fork

Perhatikan program C di bawah ini. Isilah kotak-kotak kosong berikut ini.




```
001 #include <stdio.h>
002 #include <sys/types.h>
003 #include <unistd.h>
004
005 pid_t fork_2sec() {
006   pid_t tmp;
007   sleep(1);
008   tmp = fork();
009   sleep(1);
010   return tmp;
011 }
012
013 main(){
014    int ii, jj, kk, ll;
015    ii = jj = kk = ll = (int) getpid();
016    printf("   ii     jj     kk    ll\n");
017    printf("============================\n");
018    fork_2sec();
019    jj = (int) getpid();
020    fork_2sec();
021    kk = (int) getpid();
022    fork_2sec();
023    ll = (int) getpid();
024    printf("[%3.3d] [%3.3d] [%3.3d] [%3.3d]\n",
                          ii, jj, kk, ll);
025    wait(NULL);
026    wait(NULL);
027    wait(NULL);
028 }
```

```
==========================================[============================
[     ][     ][     ][500]
[     ][     ][     ][501]
[     ][ 501][     ][502]
[     ][ 500][     ][503]
[     ][ 500][     ][504]
[     ][     ][     ][505]
[     ][     ][ 501][506]
[     ][ 500][ 500][507]
```


Revision: 716 -- 15 Nov 2017 URL: http://rms46.vLSM.org/2/183.pdf

(2011-2) Fork

Perhatikan program C di bawah ini.

```
#include <stdio.h>
#include <stdlib.h>
#include <semaphore.h>
#include "myutils.h"

int main(int argc, char * argv[]) {
   int ii;
   for (ii=0;ii<2;ii++) {
      fork();
      waitpid(-1,NULL,0);
      sleep(1);
      printf("I am %d\n",(int) getpid());
   }
}
```

Lengkapi keluaran program berikut ini:

```
I am 7000
```

(2012-1) Fork

```
01 /* main8.c (c) 2012 Rahmat M. Samik-Ibrahim  v120330
02  * getpid() = get the current PID (Process ID).
03  * fork()   = creates a new process by duplicating.
04  * wait()   = wait until one of its children terminates.
05  * GFDLike License */
06
07 #include <stdio.h>
08 #include <sys/types.h>
09 #include <sys/wait.h>
10 #include <unistd.h>
11 #define  STRING1 "PID[%5.5d] starts.\n"
12 #define  STRING2 "PID[%5.5d] passes.\n"
13 #define  STRING3 "PID[%5.5d] terminates.\n"
14
15 void main(void)
16 {
17    printf(STRING1, (int) getpid());
18    fflush(stdout);
19    for (int ii=0; ii<2; ii++) {
20       pid_t pid1=fork();
21       wait(NULL);
22       if (pid1 != 0) {
23          fork();
24          wait(NULL);
25       }
26       printf(STRING2, (int) getpid());
27    }
28    printf(STRING3, (int) getpid());
29 }
```

Lengkapi keluaran (*output*) program tersebut:

```
PID[05001] starts.
```

(2013-1) Fork

```
01 /* (c) 2013 M. Hilman and Rahmat M. Samik-Ibrahim
02  * File Name            = forkuts1301.c -- This is Free Software
03  * getpid()             = get the pid of the current process.
04  * fork()               = create/ clone a child process
05  * waitpid(-1, NULL, 0) = wait until the child process terminates
06  * ************************************************************ */
07
08 #include <sys/types.h>
09 #include <sys/wait.h>
10 #include <stdio.h>
11 #include <stdlib.h>
12 #include <unistd.h>
13 #define DISPLAY "* pid1=[%4.4d] * pid2=[%4.4d] * pid3=[%4.4d] *\n"
14 /***************************************** fork_and_report_pid */
15
16 int fork_and_report_pid (pid_t *my_pid) {
17    pid_t fork_value;
18
19    fork_value = fork();
20    *my_pid    = getpid();
21    waitpid (-1,NULL,0);
22    return  fork_value;
23 }
24 /************************************************************ main */
25 void main(void) {
26    pid_t pid1, pid2, pid3;
27
28    if (fork_and_report_pid(&pid1)) {
29       if (fork_and_report_pid(&pid2)) {
30          fork_and_report_pid(&pid3);
31       } else {
32          pid3 = getpid();
33       }
34    } else {
35       if (fork_and_report_pid(&pid2)) {
36          pid3 = getpid();
37       } else {
38          fork_and_report_pid(&pid3);
39       }
40    }
42    printf(DISPLAY, pid1, pid2, pid3);
43 }
```

Lengkapi keluaran program berikut ini:

```
* pid1=[ 5 0 0 1 ] * pid2=[     ] * pid3=[     ] *
* pid1=[          ] * pid2=[     ] * pid3=[     ] *
* pid1=[          ] * pid2=[     ] * pid3=[     ] *
* pid1=[          ] * pid2=[     ] * pid3=[     ] *
* pid1=[          ] * pid2=[     ] * pid3=[     ] *
* pid1=[          ] * pid2=[     ] * pid3=[     ] *
```


Revision: 716 -- 15 Nov 2017 URL: http://rms46.vLSM.org/2/183.pdf

(2013-2) Fork

```
 1 #include <sys/types.h>
 2 #include <sys/wait.h>
 3 #include <stdio.h>
 4 #include <unistd.h>
 5
 6 int delay1_fork (void) {
 7    sleep(1);               /* delay 1000 ms */
 8    return (int) fork();
 9 }
10
11 void main(void) {
12    int i1, i2, i3, i4, i5;
13
14    i1 = i2 = i3 = i4 = i5 = (int) getpid();
15    if (delay1_fork() == 0) {
16       i1 = (int) getpid();
17       if (delay1_fork() > 0) {
18          i2 = (int) getpid();
19          if (delay1_fork() == 0) {
20             i3 = (int) getpid();
21             if (delay1_fork() > 0) {
22                i4 = (int) getpid();
23                if (delay1_fork() == 0) {
24                   i5 = (int) getpid();
25                   sleep (1);
26                }
27             }
28          }
29       }
30    }
31   printf ("i1=%d - i2=%d - i3=%d - i4=%d - i5=%d \n", i1, i2, i3, i4, i5);
32    fflush(stdout);
33    wait(NULL);
34    wait(NULL);
35    wait(NULL);
36 }
```

a) Konversi angka ujung kanan NPM anda: **[0 → A], [1 → B], [2-3 → C], [4-5 → D], [>5 → E]** !
b) Konversi angkatan: **[<2009→I], [2009→II], [2010→III], [2011→IV], [2012→V], [2013→VI]**!
c) Harap mengisi kolom (A-E) dan baris (I – VI) hasil di atas dengan nilai "**1000**"!
d) Harap mengisi kolom dan baris lainnya sesuai dengan keluaran program di atas!

	A	B	C	D	E
I	i1=	i2=	i3=	i4=	i5=
II	i1=	i2=	i3=	i4=	i5=
III	i1=	i2=	i3=	i4=	i5=
IV	i1=	i2=	i3=	i4=	i5=
V	i1=	i2=	i3=	i4=	i5=
VI	i1=	i2=	i3=	i4=	i5=

(2014-1) Fork

a) Tuliskan NPM anda (10) digit ___ ___ ___ ___ ___ ___ ___ ___ ___ ___

b) Tuliskan digit NPM ke 8, 9, dan 10 ke baris 01, 02, dan 03 program berikut!

c) Tuliskan keluaran dari program berikut ini!

```
01  #define DIGIT08 _ /* DIGIT_08 NPM ANDA! */
02  #define DIGIT09 _ /* DIGIT_09 NPM ANDA! */
03  #define DIGIT10 _ /* DIGIT_10 NPM ANDA! */
04  #include <stdio.h>
05  #include <unistd.h>
06  #include <sys/types.h>
07  #include <sys/wait.h>

08   /* fungsi hitung:
09   * (ii + jj = genap)? return = 0; *
10   * (ii + jj = ganjil)?return = 1; */
11  int hitung(int ii, int jj) {
12     return (ii+jj)-(2*((ii+jj)/2));
13  }

15  /* false == 0  true == selain 0 */
16  void main () {
17     int ii, jj, kk;
18     ii = hitung (DIGIT08, DIGIT09);
19     jj = hitung (DIGIT08, DIGIT10);
20     kk = hitung (DIGIT09, DIGIT10);
21     if (ii)
          fork();
22     ii += jj + kk;
23     printf("PASS1 = %d\n", ii);
24     sleep(1);
25     if (jj)
          fork();
26     jj += ii + kk;
27     printf("PASS2 = %d\n", jj);
28     sleep(1);
29     if (kk)
          fork();
30     kk += ii + jj;
31     printf("PASS3 = %d\n", kk);
32  }
```

(2014-2) Fork

```
01  #include <stdio.h>
02  #include <sys/types.h>
03  #include <unistd.h>
04
05  int forkDelayChildAndAddLevel(int level) {
06     if (! fork()) {
07        level++;
08     }
09     sleep(level);
10     return level;
11  }
12
14  void main() {
15     int level = 0;
16     level=forkDelayChildAndAddLevel(level);
17     level=forkDelayChildAndAddLevel(level);
18     level=forkDelayChildAndAddLevel(level);
19     wait(NULL);
20     wait(NULL);
21     wait(NULL);
22     printf("Level[%d]: PID[%d] (PPID[%d])\n",
23                  level, getpid(), getppid());
24  }
```

Lengkapi keluaran program berikut:

```
Level[  ]: PID[      ] (PPID[      ])
Level[  ]: PID[ 1005 ] (PPID[ 1002 ])
Level[  ]: PID[      ] (PPID[      ])
Level[  ]: PID[      ] (PPID[      ])
Level[ 3]: PID[ 1007 ] (PPID[ 1004 ])
Level[  ]: PID[      ] (PPID[      ])
Level[  ]: PID[      ] (PPID[      ])
Level[  ]: PID[ 1000 ] (PPID[  999 ])
```

(2015-1) Fork

```
001 /* FORK 2015-1
002  * (c) 2015 M. Anwar Ma'sum and Rahmat M. Samik-Ibrahim
003  * This is a free software ------ Rev. 05 - 06-Apr-2015
004  */
005
006 #include <stdio.h>
007 #include <sys/types.h>
008 #include <unistd.h>
009
010 void main() {
011    pid_t  pid1, pid2, pid3;
012
013    pid1 = pid2 = pid3 = getpid();
014    printf(" EXT   MX    REG\n==================\n");
015    printf("[%4d][%4d][%4d]\n", pid1, pid2, pid3);
016    fork();
017    pid1 = getpid();
018    wait(NULL);
019    pid2 = getpid();
020    if(!fork()) {
021      pid2 = getpid();
022      fork();
023    }
024    pid3 = getpid();
025    wait(NULL);
026    printf("[%4d][%4d][%4d]\n", pid1, pid2, pid3);
027 }
028
```

a) **(KOLOM)** Beri silang kelas anda (A) **EXT**ENSI (B) **M**ATRI**X** (C) **REGULAR**

b) **(BARIS)** Lingkari sesuai angka terakhir (paling kanan) dari NPM anda **0 1 2 3 4 5 6**

c) Harap mengisi (KOLOM:BARIS) dengan 2000

d) Harap mengisi kolom dan baris lainnya sesuai dengan keluaran program di atas!

NPM	*EXT*	*MX*	*REG*
0	[ ]	[ ]	[ ]
1	[ ]	[ ]	[ ]
2	[ ]	[ ]	[ ]
3	[ ]	[ ]	[ ]
4	[ ]	[ ]	[ ]
5	[ ]	[ ]	[ ]
6	[ ]	[ ]	[ ]

Revision: 716 -- 15 Nov 2017 URL: http://rms46.vLSM.org/2/183.pdf

(2014-1) Multi-threaded Processes

Mayoritas Laptop dan Desktop masa kini dibekali dengan *multi-core processor*. Meski pun demikian, program hanya akan berjalan dalam satu *core processor* jika kode program tidak dirancang untuk dapat berjalan secara paralel dengan memanfaatkan *multi-core processor* yang ada.

Contoh Kasus

Berikut sebuah program "X" yang dilengkapi opsi untuk membagi diri secara merata menjadi satu atau lebih *thread* yang berjalan secara paralel. Dengan menggunakan utilitas "*time*" dihitung beberapa parameter berikut:

- *thread:* jumlah thread yang dibentuk
- *real:* jumlah waktu total dari "start" hingga "rampung" dalam satuan milidetik (ms).
- *user:* jumlah TOTAL (**semua thread**) CPU TIME dalam *user-mode* (ms).
- *sys*: jumlah TOTAL CPU TIME dalam *system/supervisor-mode* (ms).

THREAD	***1***	***2***	***3***	***4***	***5***
real	16154	8226	7490	6031	6204
user	15868	16100	19872	23064	22988
sys	208	208	228	220	224

a) Hitung secara kasar (bulatkan) percepatan/*speedup* yang didapatkan, jika membagi program ke dalam dua *thread*!

b) Apa yang terjadi, jika jumlah *thread* lebih besar dari pada jumlah "*core*" dan/atau "*hyperthread*"?

c) **Perkirakan, berapa jumlah "*core*" dan/atau "*hyperthread*" yang ada pada sistem ini? Terangkan!**

(2011-1) Sinkronisasi

Perhatikan berkas header “myutils.h” berikut ini:

```
#define MAX_THREAD 100
#define BUFFER_SIZE  5
#define TRUE         1
#define FALSE        0
typedef struct {
   int   buffer[BUFFER_SIZE];
   int   in;
   int   out;
   int   count;
} bbuf_t;
void daftar_trit   (void* trit);      // mempersiapkan "trit"
void jalankan_trit (void);            // menjalankan dan menunggu hasil dari
                                      // "daftar_trit"
void beberes_trit  (char* pesan);     // beberes menutup "jalankan_trit"
void rehat_acak    (long max_mdetik); // istirohat acak "0-max_mdetik" (ms)
void init_buffer   (void);            // init buffer
void enter_buffer  (int entry);       // enter  an integer item
int remove_buffer  (void);            // remove the item
void init_rw       (void);            // init readers writers
int  startRead     (void);            // start reading
int  endRead       (void);            // end reading
void startWrite    (void);            // start writing
void endWrite      (void);            // end writing
```

Apabila fungsi "**rand()**" baris 29 menghasilkan nilai "25", bagaimana bentuk keluaran program berikut ini?

```
001 #include <stdio.h>
002 #include <stdlib.h>
003 #include <semaphore.h>
004 #include <time.h>
005 #include "myutils.h"
006 #define  MAX_TRIT 10
007 int      nomor_trit=0;
008 sem_t    mutex;
009 sem_t    tmutex[MAX_TRIT];
010 void init_random() {
011   srand(time(NULL));
012 }
013 void* TRIT_sederhana (void* a) {
014    //ID dari trit yang sedang berjalan.
015    int trit_ID;
016    sem_wait (&mutex);
017    trit_ID  = nomor_trit++;
018    sem_post (&mutex);
019    sem_wait (&tmutex[trit_ID]);
020    printf("TRIT No %d\n", trit_ID);
021    int next = trit_ID + 1;
022    if (next >= MAX_TRIT)
023       next = next % MAX_TRIT;
024    sem_post (&tmutex[next]);
025 }
026 int main(int argc, char * argv[]){
027    int   ii;
028    init_random();
029    int mulai = rand() % MAX_TRIT;
030    for (ii=0; ii < MAX_TRIT; ii++) {
031       sem_init   (&tmutex[ii], 0, 0);
032       daftar_trit(TRIT_sederhana);
033    }
034    sem_init (&mutex, 0, 1);
035    sem_post (&tmutex[mulai]);
036    jalankan_trit();
037    beberes_trit("INDUK mohon diri");
038 }
039  /* DICONTEK DAN DIMODIF DARI B210  */
```

(2011-2) Sinkronisasi

Perhatikan berkas header “myutils.h” berikut ini:

```
#define MAX_THREAD 100
#define BUFFER_SIZE  5
#define TRUE         1
#define FALSE        0
typedef struct {
   int   buffer[BUFFER_SIZE];
   int   in;
   int   out;
   int   count;
} bbuf_t;
void daftar_trit   (void* trit);      // mempersiapkan "trit"
void jalankan_trit (void);            // menjalankan dan menunggu hasil dari
                                      // "daftar_trit"
void beberes_trit  (char* pesan);     // beberes menutup "jalankan_trit"
void rehat_acak    (long max_mdetik); // istirohat acak "0-max_mdetik" (ms)
void init_buffer   (void);            // init buffer
void enter_buffer  (int entry);       // enter  an integer item
int remove_buffer  (void);            // remove the item
void init_rw       (void);            // init readers writers
int  startRead     (void);            // start reading
int  endRead       (void);            // end reading
void startWrite    (void);            // start writing
void endWrite      (void);            // end writing
```

a) Tuliskan keluaran dari program berikut ini:

```
#include <stdio.h>
#include <stdlib.h>
#include <semaphore.h>
#include "myutils.h"
sem_t                   mutex1, mutex2;

void* TRIT_satu (void* a) {
   sem_wait (&mutex1);
   printf("Ini TRIT satu\n");


}

void* TRIT_dua (void* a) {
   sem_wait (&mutex2);
   printf("Ini TRIT dua\n");
   sem_post (&mutex1);
}

void* TRIT_tiga (void* a) {
   printf("Ini TRIT tiga\n");
   sem_post (&mutex2);
}


int main(int argc, char * argv[]) {
   sem_init (&mutex1, 0, 0);
   sem_init (&mutex2, 0, 0);


   daftar_trit(TRIT_satu);
   daftar_trit(TRIT_dua);
   daftar_trit(TRIT_tiga);


   jalankan_trit();
   beberes_trit("INDUK mohon diri");
}
```

_____

_____

_____

_____

_____

_____

b) Tambahkan thread “TRIT_empat” yang akan mencetak setelah “TRIT_satu”. Lengkapi bagian yang kosong pada program sebelah kiri ini.

(2012-1) Sinkronisasi

```
01  /*
02   * $Revision: 140 $
03   * (c) 2012 Rahmat M. Samik-Ibrahim
04   * This is FREE SOFTWARE.
05   * myutils.c/myutils.h provides:
06   *   rehat_acak(), daftar_trit(),
07   *   jalankan_trit(), beberes_trit()
08   */
09
10  #include <stdio.h>
11  #include <stdlib.h>
12  #include <semaphore.h>
13  #include "myutils.h"
14
15  #define      lamaRehat  500
16  #define      jmlPembalap  3
17  sem_t        mutex, start;
18
19  void* bandar (void* a) {
20     for (int ii=0; ii<jmlPembalap; ii++)
21        sem_wait  (&start);
22     sem_wait  (&mutex);
23     rehat_acak(lamaRehat);
24     printf    ("Bandar Siap!\n");
25     sem_post  (&mutex);
26  }
27
28  int   idmaster = 1;
29  int   juara    = 1;
30  int   menang   = TRUE;
31  void* pembalap (void* a) {
32     int id;
33     sem_wait  (&mutex);
34     id        = idmaster++;
35     sem_post  (&mutex);
36     sem_post  (&start);
37     printf    ("Pembalap Siap!\n");
38     rehat_acak(lamaRehat);
39     sem_wait  (&mutex);
40     if  (menang==TRUE) printf("HORE, pemain");
41     else               printf("Aduh, pemain");
42     printf(" %2.2d juara %2.2d!\n",id,juara++);
43     menang    = FALSE;
44     sem_post  (&mutex);
45  }
46
47  void main(void) {
48     sem_init (&mutex, 0, 1);
49     sem_init (&start, 0, 0);
50     daftar_trit     (bandar);
51     for (int ii=0; ii<jmlPembalap; ii++)
52        daftar_trit  (pembalap);
53     jalankan_trit   ();
54     beberes_trit    ("Selese...");
55  }
```

a) Ada berapa thread "pembalap" yang akan "balapan"? Sebutkan "id" dari semua thread "pembalap" tersebut!
b) Sebutkan nomor baris-baris program yang merupakan *critical section*. Sebutkan juga variabel penyebab terjadinya *critical section* tersebut. Terakhir, sebutkan nama semaphore yang "menjaga" *critical section* tersebut!
c) Sebutkan secara singkat peranan dari *semaphore* "start".
d) Fungsi "rehat_acak()" (baris 35) berturut-turut memberikan "rehat"/ waktu tunda selama: 400 ms, 20ms, dan 150ms. Tentukan keluaran (output) dari program tersebut:
e) Modifikasi program, sehingga jumlah pembalap menjadi 50. Sebutkan, baris mana saja yang mesti dimodifikasi

(2012-2) Sinkronisasi

Tambahkan ***pseudo program*** pada soal "**(2012-2) Penjadwalan Thread**" dengan semafor, agar **urutan** eksekusi ***thread*** seperti yang diinginkan. Tambahkan pada program tersebut, kombinasi dari ketiga fungsi semafor berikut ini:
sem_init(X,Y) – inisialisasi semafor "X" dengan nilai "Y".
sem_wait(X) – fungsi wait semafor "X".
sem_signal(X) – fungsi signal semafor "X".

(2013-1) Sinkronisasi

Pseudo program **(T1, T2, T3, T4, T5)** berikut bertujuan untuk menghitung lima (5) elemen pertama dari deret ***Fibonacci*** (F1, F2,... F5).

a) Tambahkan sinkronisasi **semafor** pada program tersebut dengan menggunakan kombinasi dari ketiga fungsi berikut ini:
   `sem_init(X,Y)` – inisialisasi semafor "X" dengan nilai "Y".
   `sem_wait(X)` – fungsi wait semafor "X".
   `sem_signal(X)` – fungsi signal semafor "X".

```
Main/T1:
   F0 = 0
   F1 = 1

T2:

   F2 = F1 + F0

T3:
   F3 = F2 + F1

T4:
   F4 = F3 + F2

T5:
   F5 = F4 + F3
```

b) Berikan komentar/tunjukan mengapa deret ***Fibonacci*** disebut problem yang tidak dapat diparalelkan.

(2013-2) Sinkronisasi

a) Perhatikan program pada halaman berikut. Program tersebut menggunakan fungsi "daftar_trit()" untuk mendaftar "threat" dan "jalankan_trit()" untuk menjalankan "threat". Fungsi "sem_init()" untuk inisialisasi semaphore, "sem_post()" untuk signal semaphore, dan "sem_wait()" untuk wait semaphore. Bagaimana salah satu kemungkinan keluaran dari program tersebut?

b) Dari keluaran program di atas, baris berapa saja yang mungkin keluar dengan urutan yang berbeda? Jelaskan mengapa!

c) Apa peranan semaphore "mutex" pada program tersebut?

d) Apa peranan semaphore "switch1" pada program tersebut?

e) Apa peranan semaphore "switch2" pada program tersebut?

```
 1 /*
 2  * $Revision: 140 $
 3  * (c) 2013 Rahmat M. Samik-Ibrahim
 4  * This is FREE SOFTWARE.
 5  */
 6
 7 #include <stdio.h>
 8 #include <stdlib.h>
 9 #include <semaphore.h>
10 #include "myutils.h"
11
12 #define      NThreads 4
13
14 sem_t        mutex,   switch1, switch2;
15 int          addvar1, addvar2, addresult;
16 int          subvar1, subvar2, subresult;
17 int          mulvar1, mulvar2, mulresult;
18 int          divvar1, divvar2, divresult;
19
20 void* manager (void* a) {
21    printf("Manager starts \n");
22
23    for (int ii=0; ii< NThreads;ii++)
24        sem_wait (&switch1);
25    sem_wait  (&mutex);
26    addvar1 = 5;
27    addvar2 = 2;
28    subvar1 = 7;
29    subvar2 = 2;
30    mulvar1 = 2;
31    mulvar2 = 3;
32    divvar1 = 4;
33    divvar2 = 2;
34    sem_post  (&mutex);
35
36    for (int ii=0; ii< NThreads;ii++)
37        sem_post (&switch2);
38    for (int ii=0; ii< NThreads;ii++)
39        sem_wait (&switch1);
40    printf("Result add:%d; sub:%d; mul:%d;
         div:%d;\n",
41       addresult, subresult, mulresult,
         divresult);
42 }
43
44 void* add (void* a) {
45    sem_post  (&switch1);
46    sem_wait  (&switch2);
47
48    sem_wait  (&mutex);
49    printf("Add starts \n");
50    addresult = addvar1 + addvar2;
51    sem_post  (&mutex);
52    sem_post  (&switch1);
53 }
54
55 void* subtract (void* a) {
56    sem_post  (&switch1);
57    sem_wait  (&switch2);
58
59    sem_wait  (&mutex);
60    printf("Subtract starts \n");
61    subresult = subvar1 - subvar2;
62    sem_post  (&mutex);
63    sem_post  (&switch1);
64 }
65
66 void* multiply (void* a) {
67    sem_post  (&switch1);
68    sem_wait  (&switch2);
69    sem_wait  (&mutex);
70    printf("Multiply starts \n");
71    mulresult = mulvar1 * mulvar2;
72    sem_post  (&mutex);
73    sem_post  (&switch1);
74 }
75
76 void* divide (void* a) {
77    printf("Divide starts \n");
78    sem_post  (&switch1);
79    sem_wait  (&switch2);
80    sem_wait  (&mutex);
81    divresult = divvar1 / divvar2;
82    sem_post  (&mutex);
83    sem_post  (&switch1);
84 }
85
86 void main(void) {
87    sem_init      (&mutex,   0, 1);
88    sem_init      (&switch1, 0, 0);
89    sem_init      (&switch2, 0, 0);
90    daftar_trit   (manager);
91    daftar_trit   (add);
92    daftar_trit   (subtract);
93    daftar_trit   (multiply);
94    daftar_trit   (divide);
95    jalankan_trit ();
96    beberes_trit  ("Done...");
97 }
```

(2014-2) Sinkronisasi

```
001  // R01--19-OCT-2014
002  // (c) 2014 Rahmat M. Samik-Ibrahim
003  // This is FREE SOFTWARE.
005  // *Rock*Paper*Scissors*Lizard*Spock*
006  // Invented by Sam Kass and Karen Bryla
007  // Rock crushes Scissors
008  // Rock crushes Lizard
009  // Paper covers Rock
010  // Paper disproves Spock
011  // Scissors cut Paper
012  // Scissors decapitate Lizard
013  // Lizard eats Paper
014  // Lizard poisons Spock
015  // Spock vaporizes Rock
016  // Spock smashes Scissors
018  #include <semaphore.h>
019  #include <stdio.h>
020  #include <stdlib.h>
021  #include <time.h>
022  #include <unistd.h>
023  #include "myutils.h"
025  #define nPlayers 2
026  #define nWeapons 5
027  int     playerSEQ=1;
028  int     myWeapon[nPlayers];
029  sem_t   mutex, sync1, sync2;
031  // (0=Rock) (1=Paper) (2=Scissors)
032  // (3=Lizard) (4=Spock)
033  char *weaponName[nWeapons]= {
034     "Rock", "Paper", "Scissors",
035     "Lizard", "Spock"
036  };
037
038  // '-' = draw  'v' = win  'x' = lose
039  char weaponTable[nWeapons][nWeapons] = {
040     {'-','x','v','v','x'},
041     {'v','-','x','x','v'},
042     {'x','v','-','v','x'},
043     {'x','v','x','-','v'},
044     {'v','x','v','x','-'}
045  };
046
047  void waitPlayers() {
048     for (int ii=0; ii < nPlayers; ii++)
049        sem_wait(&sync1);
050  }
051
052  void postPlayers() {
053     for (int ii=0; ii < nPlayers; ii++)
054        sem_post(&sync2);
055  }
056
057  void* playerThread (void* a) {
058     int      playerID;
059     sem_wait (&mutex);
060     playerID=playerSEQ++;
061     sem_post (&mutex);
062     printf("Player[%d]: READY\n",playerID);
063     sem_post (&sync1);
064     sem_wait (&sync2);
065     myWeapon[playerID] = rand() % nWeapons;
066     printf("Player[%d]: %s\n", playerID,
067        weaponName[myWeapon[playerID]]);
068     sem_post (&sync1);
069  }
```

```
071  void* refereeThread (void* a) {
072     waitPlayers();
073     printf("Referee:   ALL READY!\n");
074     postPlayers();
075     waitPlayers();
076     char result =
077        weaponTable[myWeapon[1]][myWeapon[2]];
078     if (result == '-')
079        printf("Referee:  DRAW!\n");
080     else if (result == 'v')
081        printf("Referee:  Player[1] WINS!\n");
082     else
083        printf("Referee:  Player[2] WINS!\n");
084  }
085
086  void main() {
087     // randomize with a time seed
088     srand(time(NULL));
089     sleep(1);
090     // init semaphore mutex = 1 syncx = 0
091     sem_init (&mutex, 0, 1);
092     sem_init (&sync1, 0, 0);
093     sem_init (&sync2, 0, 0);
094     // register and execute threads
095     daftar_trit (refereeThread);
096     for (int ii=0; ii<nPlayers; ii++)
097        daftar_trit (playerThread);
098     jalankan_trit ();
099     beberes_trit  ("Goodbye...");
100  }
```

a) Tuliskan NPM anda (10 dijit):

___ ___ ___ ___ ___ ___ ___ ___ ___ ___

b) Player #1: fungsi “rand()” baris 65 dihasilkan dari gabungan digit NPM #7 + #8 yaitu:

___ ___

c) Player #1: nilai “myWeapon[1]” ialah: _____

d) Player #1: “weaponName” ialah: _____

e) Player #2: fungsi “rand()” baris 65 dihasilkan dari gabungan digit NPM #9 + #10 yaitu:

___ ___

f) Player #2: nilai “myWeapon[2]” ialah: _____

g) Player #2: “weaponName” ialah: _____

h) Sebutkan baris mana pada program yang merupakan “critical section”!

i) Tuliskan baris-baris keluaran program tersebut.

(2015-1) Sinkronisasi

```
001 /* SSV: Sudoku Solution Validator
002  * (c) 2015 M. Anwar Ma'sum and R.M. Samik-Ibrahim
003  * This is a free software - Rev. 05 - 06-Apr-2015
004  */
005
006 #include <stdio.h>
007 #include <pthread.h>
008 #include <semaphore.h>
009 #include "myutils.h"
010 #define  V_THREADS 27
011
012 int   idSequence = 0;
013 sem_t mutex, sync;
014 char  result[3][9];
015 int   sudoku[9][9] = {   /* Check this 9x9 matrix
*/
016    {5,3,4, 7,6,8, 9,1,2},
017    {6,7,2, 1,9,5, 3,4,8},
018    {1,9,8, 3,4,2, 5,6,7},
019
020    {8,5,9, 6,7,1, 4,2,3},
021    {4,2,6, 8,5,3, 7,9,1},
022    {7,1,3, 9,2,4, 8,5,6},
023
024    {9,6,1, 5,3,7, 2,8,4},
025    {2,8,7, 4,1,9, 6,3,5},
026    {3,4,5, 2,8,6, 1,7,8}
027 };
028
029 char validate(int iINIT,int iEND,int jINIT,int
jEND)
030 {
031    int  ii, jj;
032    char flag[9];
033
034    for (ii = 0; ii < 9; ii++) flag[ii] = 'F';
035    for (ii = iINIT; ii < iEND; ii++) {
036       for (jj = jINIT; jj < jEND; jj++) {
037          if (flag[sudoku[ii][jj]-1] == 'F')
038              flag[sudoku[ii][jj]-1] =  'T';
039          else
040              return 'F';
041       }
042    }
043    return 'T';
044 }
045
046 void *sudokuValidator (void *param) {
047     int my_ID, tmp0, tmp1;
048     char check;
049
050     sem_wait(&mutex);
051     my_ID = idSequence++;
052     sem_post(&mutex);
053
054     if (my_ID < 9) {
055         check = validate (my_ID, my_ID+1, 0, 9);
056     } else if (my_ID < 18) {
057         check = validate (0,9,my_ID%9,my_ID%9+1);
058     } else {
059         tmp0  = ((my_ID%9)/3)*3;
060         tmp1  = ((my_ID%9)%3)*3;
061         check = validate (tmp0,tmp0+3,tmp1,tmp1+3);
062     }
063
064     sem_wait(&mutex);
065     result[(my_ID/9)][(my_ID%9)] = check;
066     sem_post(&mutex);
067     sem_post(&sync);
068 }
069
070 void *reporter (void *p) {
071    int ii,jj;
072    for (ii = 0; ii < V_THREADS; ii++)
073       sem_wait(&sync);
074    for (ii = 0; ii < 3; ii++) {
075       if      (ii == 0) printf ("ROW Validators: ");
076       else if (ii == 1) printf ("COL Validators: ");
077       else              printf ("BOX Validators: ");
078       for (jj = 0; jj < 9; jj++)
079          printf("%c ", result[ii][jj]);
080       printf("\n");
081    }
082 }
083
084 void main(void *v) {
085   int ii;
086   printf("SSV Sudoku Solution Validator\n");
087   sem_init(&mutex,0,1);
088   sem_init(&sync, 0,0);
089   daftar_trit(reporter);
090   for (ii = 0; ii < V_THREADS; ii++)
091       daftar_trit(sudokuValidator);
092   jalankan_trit();
093   beberes_trit("Done...");
094 }
```

Program SSV *(**Sudoku Solution Validator**)* ini menggunakan pustaka *(library*) **PTHREAD** dan **MYUTILS**. SSV melakukan pemeriksaan terhadap 9 “baris” (**RO**W), 9 “kolom” (**COL**), dan 9 “kotak 3x3” (**BOX**).

a) Tentukan keluaran (***output***) dari program tersebut!

b) Berapa jumlah total “***thread***” yang terbentuk setelah “**`jalankan_trit()`**” (baris 92)?

c) Lingkari sesuai angka terakhir (paling kanan) dari NPM anda **0 1 2 3 4 5 6**

d) Tentukan parameter **`“iINIT”, “iEND”, “jINIT”, “jEND”`**, jika ingin validasi kolom (**COL**) dengan nomor “**ujung NPM anda**” untuk fungsi pada baris 29 yaitu:
`validate(int iINIT, int iEND, int jINIT, int jEND)`

e) Apa peranan semafor “mutex” pada baris 50 dan 52? Apa yang dapat terjadi, jika baris 52 dan baris 50 dihapus? Terangkan!

f) Apa peranan semafor “mutex” pada baris 64 dan 66? Apa yang dapat terjadi, jika baris 66 dan baris 64 di hapus? Terangkan!

g) Apa peranan semafor “sync” pada baris 67? Apa yang dapat terjadi, jika baris 67 tersebut di hapus? Terangkan!

h) Apa peranan semafor “sync” pada baris 73? Apa yang dapat terjadi, jika baris 73 tersebut di hapus? Terangkan!

i) Program ini memiliki keterbatasan yaitu angka dalam matrix sudoku diasumsikan antara 1 hingga 9. Sebutkan problem yang dapat timbul jika ada nilai dalam matrix yang diluar 1-9 ?

(2012-2) *Deadlock*

a) Sebutkan dengan singkat, keempat kondisi yang dapat mengakibatkan “*deadlock*”! Terangkan setiap kondisi dalam 1-2 kalimat.
b) Terangkan dengan singkat, ketiga jenis upaya untuk menangani “*deadlock*”! Terangkan setiap upaya, dalam 1-2 kalimat.
c) Dari ketiga jenis upaya butir (b) di atas, upaya mana yang paling lazim dilaksanakan? Berikan alasan dalam 1-2 kalimat.

(2011-1/Silberschatz) Memori

Diketahui sebuah reference string “1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6”. Lengkapilah tabel JUMLAH PAGE FAULT untuk algoritma-algoritma berikut ini:

Jumlah Frames	LRU	FIFO	Optimal
1			
2			
3			
4			
5			
6			
7			

(2012-2) Memori

Diketahui sebuah sistem dengan 32-bit VA dan 64-bit PA. Ukuran halaman (page size) ialah 64 k-bytes, serta semua notasi dalam notasi Hexadesimal. Setiap baris TLB terdiri dari VPN, PFN, dan a valid bit (valid=1).

TLB (Hexadecimal)		
VPN	**PFN**	**Valid**
F	123456789ABC	1
FE	123456789AB	1
FED	123456789A	0
FEDC	123456789	1

a) Lengkapi skema VA berikut ini; tentukan berapa bit untuk VPN dan berapa bit untuk offset.

b) Berapakah ukuran bingkai (frame) dalam k-bytes?

c) Lengkapi skema PA berikut ini; tentukan berapa bit untuk PFN dan berapa bit untuk offset.

d) Apakah PA dari VA berikut ini:

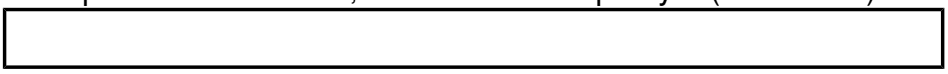
Address Translation		
VA	*Valid (Yes/No)*	*PA*
FEDCBA98		
FEDCBA9		
FEDCBA		
FEDCB		

(2011-2/UCB Fall 2008) Memori

Diketahui sistem alamat 32-bit virtual dengan alamat fisik 2 TB. Ukuran halaman (page size) ialah 8 kB. Setiap "Page Table Entry" (PTE) akan terdiri dari:

- beberapa **bits** untuk bingkai alamat fisik;
- satu Valid/Invalid **bit;** dan
- beberapa **bit**s kosong.

a) Berapa ukuran bingkai (*frame size*)? (1 TB = 1024 GB = 1024 x 1024 MB)
b) Berapa bingkai (*frame)* yang diperlukan untuk menampung seluruh alamat fisik tersebut?
c) Berapa bit yang diperlukan dalam PTE untuk merepresentasikan ukuran frame?
d) Gambarkan skema PTE lengkap yang terdiri dari bit "frame numbers", bit "valid/invalid", serta beberapa bit. Secara total, dibutuhkan berapa byte (dibulatkan)?



e) Berapa PTEs dapat muat dalam satu halaman virtual?
f) Gambarkan skema alamat virtual lengkap dengan jumlah masing-masing bit "page numbers" dan "offset".

g) Berapa PTEs yang dibutuhkan untuk memetakan seluruh ruang alamat virtual tersebut?
h) Berapa total jumlah halaman virtual yang diperlukan untuk menampung seluruh PTEs di atas?

(2012-1/UCB Fall 2010) Memori

Diketahui alamat virtual dua tingkat 24 bit, dengan alamat fisik 16 bit.

PT1 Index (8 bit)	PT2 Index (8 bit)	Offset (8 bit)

Penempatan PTE (Page Table Entry) pada memori fisik dalam format 16 bit BIG ENDIAN: 8 bit pertama untuk nomor bingkai (frame) dan 8 bit berikutnya untuk flag diantaranya Valid/Invalid (1/0), Dirty/Clean (1/0), dst.

Page Table Entry (PTE)

Physical Page # (8 bits)	Kernel Only	Uncacheable	0	0	Dirty	Use	Write	Valid

Table base pointer/Page Table Base Register berisikan lokasi awal PT1 dengan alamat 0x2000.

a) Berapa byte ukuran halaman (page)?

b) Berapa byte ukuran bingkai (frame)?

c) Berapa byte ukuran total PT1? Berapa byte ukuran total dari sebuah PT2?

d) Berapa byte ukuran total seluruh PT2?

e) Berapa jumlah bit pada satu entri TLB?

f) Buat diagram lengkap translasi pengalamatan alamat virtual dengan PT1, sebuah PT2, TLB, *page table base register* dan alamat fisik.

g) Pada tabel halaman 3, terdapat alamat dan isi memori fisik dalam format heksadesimal. Tentukan valid atau invalid alamat virtual dibawah ini. Jika valid, tentukan isinya. Jika invalid, tentukan invalid terjadi pada PT1 atau PT2

   i. 0x0700FE

   ii. 0x0C2345

   iii. 0x000115

   iv. 0x080D09

PT1 Index (8 bit)	PT2 Index (8 bit)	Offset (8 bit)

Page Table Entry (PTE)

Physical Page # (8 bits)	Kernel Only	Uncacheable	0	0	Dirty	Use	Write	Valid

Physical Memory

Address	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0x0000	E0	F0	01	11	21	31	41	51	61	71	81	91	A1	B1	C1	D1
0x0010	1E	1F	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D
....																
0x1010	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
0x1020	40	07	41	06	30	06	31	07	00	07	00	00	00	00	00	00
....																
0x2000	21	01	22	03	25	01	22	01	2F	03	28	03	30	03	22	03
0x2010	40	81	41	81	42	81	43	83	00	00	00	00	00	00	00	00
....																
0x2100	30	05	31	01	32	03	33	07	34	00	35	00	36	00	37	00
0x2110	38	00	39	00	3A	00	3B	00	3C	00	3D	00	3E	00	3F	00
....																
0x2200	30	01	31	83	00	01	00	0F	04	00	05	00	06	00	07	00
0x2210	08	00	09	00	0A	00	0B	00	0C	00	0D	00	0E	00	0F	00
....																
0x2500	10	01	00	03	12	85	13	05	14	05	15	05	16	05	17	05
0x2510	18	85	19	85	1A	85	1B	85	1C	85	1D	85	1E	85	00	00
....																
0x2800	50	01	51	03	00	00	00	00	00	00	00	00	00	00	00	00
....																
0x2F00	60	03	28	03	62	00	63	00	64	03	65	00	66	00	67	00
0x2F10	68	00	69	00	00	00	00	00	00	00	00	00	00	00	00	00
0x2F20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
....																
0x30F0	00	11	22	33	44	55	66	77	88	99	AA	BB	CC	DD	EE	FF
0x3100	01	12	23	34	45	56	67	78	89	9A	AB	BC	CD	DE	EF	00
0x3110	02	13	24	35	46	57	68	79	8A	9B	AC	BD	CE	DF	F0	01
....																
0x4000	30	00	31	06	32	07	33	07	34	06	35	00	43	38	32	79
0x4010	50	28	84	19	71	69	39	93	75	10	58	20	97	49	44	59
0x4020	23	87	20	07	00	06	62	08	99	86	28	03	48	25	34	21

(2013-1) Memori

Diketahui sebuah sistem memori SATU TINGKAT dengan alamat virtual 16 bit serta alamat fisik 16 bit. Setiap PTE (*Page Table Entry*) berisi nomor bingkai/*frame* (8 bit) dari memori fisik. Jika (PTE == 00), maka PTE dinyatakan **tidak sah** (*invalid*); sedangkan jika (PTE !=00), maka PTE dinyatakan **sah** (*valid*). Alamat awal PT (*Page Table*) ialah memori fisik 1000 (HEX).

Bagan alamat virtual:

Page Table Index (8 bits)	**Offset (8 bits)**

PTE (Page Table Entry)

Physical Frame Number (8 bits)

Physical Memory

Address (HEX)	***+0***	***+1***	***+2***	***+3***	***+4***	***+5***	***+6***	***+7***	***+8***	***+9***	***+A***	***+B***	***+C***	***+D***	***+E***	***+F***
1000	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
1010	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
.....																
2000	00	01	00	02	00	03	00	04	00	05	00	06	00	07	00	08
.....																
2100	00	11	00	12	00	13	00	14	00	15	00	16	00	17	00	18
.....																
3000	00	00	00	01	00	02	FE	FD	FC	FB	FA	F9	F8	F7	F6	F5
.....																
3100	00	02	00	04	00	06	EE	ED	EC	EB	EA	E9	E8	E7	E6	E5
.....																

a) Berapa byte ukuran sebuah halaman (*page*)?

b) Berapa byte ukuran sebuah bingkai (*frame*)?

c) Maksimum, akan ada berapa PTE (*Page Table Entries*) dalam PT (*Page Table*) tersebut?

d) Berapa jumlah bit yang diperlukan sebuah TLB (*Translation Lookaside Buffer) entry*? Jelaskan!

e) Tentukan **alamat fisik** dari **alamat virtual** (HEX) berturut-turut berikut:
`1000, 1001, 1002, 1003, 1100, 1101, 1102, 1103.`

Revision: 716 -- 15 Nov 2017 URL: http://rms46.vLSM.org/2/183.pdf

f) Tuliskan digit NPM anda ke kotak-kotak berikut ini:

Buntut/Nomor paling kanan NPM anda ialah: _____

g) Konvensi pengalamatan berikut ini dikenal dengan nama "**BIG ENDIAN**"
Satuan dari isi alamat memori ialah *byte* atau 8 bit. Umpamanya, isi data "8 bit" dari **alamat memori fisik** [1000 HEX] ialah "20" (lihat tabel memori fisik halaman sebelumnya). Untuk ukuran data 16 bit, memerlukan dua buah alamat memori berurutan. Umpamanya, isi data "16 bit" dari **alamat memori fisik** [1000 HEX] ialah "2021". Byte "20" merupakan isi alamat fisik [1000 HEX], sedangkan byte "21" merupakan isi alamat fisik [1001 HEX]. Kerjakan operasi "16 bit" berikut sesuai dengan buntut NPM anda:

i. Untuk buntut **NPM "0" atau "1"**: hitung isi **alamat virtual** [1000] setelah operasi (HEX):

```
[1000] <--- [1000] + [1002]
```

ii. Untuk buntut **NPM "2" atau "3"**: hitung isi **alamat virtual** [1000] setelah operasi (HEX):

```
[1000] <--- [1002] - [1000]
```

iii. Untuk buntut **NPM "4" atau "5"**: hitung isi **alamat virtual** [1100] setelah operasi (HEX):

```
[1100] <--- [1100] * [1102]
```

iv. Untuk buntut **NPM "6" atau lebih**: hitung isi **alamat virtual** [1100] setelah operasi (HEX):

```
[1100] <--- [1102] / [1100]
```

(2013-2) Memori

Diketahui sebuah sistem alamat virtual (VA) 16 bit dan alamat fisik (PA) 24 bit. *Offset/displacement* ialah 12 bit. Alamat fisik *Page Table* ialah 001000 (HEX). Berikut bagan memori fisik:

Address (HEX)	***+0***	***+1***	***+2***	***+3***	***+4***	***+5***	***+6***	***+7***	***+8***	***+9***	***+A***	***+B***	***+C***	***+D***	***+E***	***+F***
001000	10	24	10	23	10	22	10	21	10	20	10	1F	10	1E	10	1D
001010	10	34	10	33	10	32	10	31	10	30	10	2F	10	2E	10	2D
.....																
01F000	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
.....																
020000	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
.....																
023000	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
.....																
031000	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
.....																

Diketahui **TLB** dengan *entry* dalam bentuk DIGIT HEX: "`[Page Number][Flags 4 bits][Frame Number]`". "*Frame Number*" TIDAK VALID jika nilai FLAGS bukan "1 (HEX)" atau "0001 (BIN)".

`[Page#]`	`[Flags]`	`[Frame#]`		
0	1	0	2	4
1	1	0	2	3
2	1	0	2	2
3	1	0	2	1
4	1	0	2	0

a) Tuliskan digit NPM anda ke kotak-kotak berikut ini:

b) Angka paling kanan NPM anda: _____

c) Tentukan VA1 dengan menambahkan 1000 (HEX) pada hasil "b" di atas: VA1 = ____ + 1000 (HEX) = ____

d) Tentukan VA2 dengan menambahkan 5000 (HEX) pada hasil "b" di atas: VA2 = ____ + 5000 (HEX) = ____

e) Tentukan komposisi bit dari *Page Number* pada bagan VA berikut ini:

Page Number (?? bit)	Offset (12 bit)

Jumlah bit yang digunakan dalam *"Page Number"* ialah _____ bit

f) Jumlah halaman (*page*) berdasarkan jumlah bit tersebut di atas ialah: _____ halaman

g) Sistem alamat fisik 24 bit menggunakan bagan sebagai berikut:

Frame Number (?? bit)	Offset (12 bit)

Jumlah bit yang digunakan dalam "*Frame Number*" ialah _____ bit

h) Jumlah bingkai (*frame*) berdasarkan jumlah bit tersebut di atas ialah: _____ bingkai

i) Sebutkan *Page Number* dari VA1 (lihat butir "c" di atas): _____

j) Apakah menemukan *Frame Number* dari VA1 di TLB ? Ya / Tidak (lingkari yang betul)

k) Apakah menemukan *Frame Number* dari VA1 di PT (*Page Table*)?Ya / Tidak

l) Status *Frame Number* untuk VA1: Valid / Tidak Valid / Tidak Ada

m) Jika valid, berapakan nilai *Frame Number* untuk VA1? _____

n) (JIKA ADA) Apakah alamat fisik (PA1) dari VA1? _____

o) Kesimpulan: (JIKA ADA) Apakah isi dari VA1? _____

p) Sebutkan *Page Number* dari VA2 (lihat butir "d" di atas): _____

q) Apakah menemukan *Frame Number* dari VA2 di TLB ? Ya / Tidak

r) Apakah menemukan *Frame Number* dari VA2 di PT (*Page Table*)?Ya / Tidak

s) Status *Frame Number* untuk VA2: Valid / Tidak Valid / Tidak Ada

t) Jika valid, berapakan nilai *Frame Number* untuk VA2?_____

u) (JIKA ADA) Apakah alamat fisik (PA2) dari VA2?_____

v) Kesimpulan: (JIKA ADA) Apakah isi dari VA2? _____

(2014-1) Memori

a) Tulis NPM (Nomor Pokok Mahasiswa) anda:

b) Tulis ujung kanan (buntut) NPM anda:

BUNTUT = | |

c) Konversikan nilai BUNTUT dari DESIMAL ke **INTEGER-32BIT** HEXADESIMAL dengan menambahkan "0" secukupnya:

								(HEX)

Diketahui sebuah sistem alamat virtual (VA) 32 bit dengan alamat fisik (PA) 32 bit. *Displacement*/ *Offset* ialah 16 bit.

d) Beri label serta lengkapi bagan Alamat Virtual (VA) dari sistem ini:

Komponen Nomor Halaman (PAGE #) = _____ bit = _____ digit (HEX)

Komponen Offset = _____ bit = _____ digit (HEX)

BAGAN VA (HEX):

e) Beri label serta lengkapi bagan Alamat Fisik (PA) dari sistem ini:

Komponen Nomor Bingkai (FRAME #) = _____ bit = _____ digit (HEX)

Komponen Offset = _____ bit = _____ digit (HEX)

BAGAN PA (HEX):

Tambahkan BUNTUT (butir c) ke alamat-alamat virtual berikut:

f) Hitung Alamat Virtual 1 (VA1): VA1 = 0000 1000 (HEX) + BUNTUT (HEX) =

								(HEX)

g) Hitung Alamat Virtual 2 (VA2): VA2 = 0000 2000 (HEX) + BUNTUT (HEX) =

								(HEX)

h) Hitung Alamat Virtual 3 (VA3): VA3 = 0001 0000 (HEX) + BUNTUT (HEX) =

								(HEX)

Alamat fisik *Page Table* (PT) ialah 00001000 (HEX). Berikut bagan **BIG ENDIAN** memori PT:

Address (HEX)	***+0***	***+1***	***+2***	***+3***	***+4***	***+5***	***+6***	***+7***	***+8***	***+9***	***+A***	***+B***	***+C***	***+D***	***+E***	***+F***
0000 1000	01	23	45	67	89	AB	CD	EF	01	23	45	67	89	AB	CD	EF
0000 1010	01	23	45	67	89	AB	CD	EF	01	23	45	67	89	AB	CD	EF

Baik PTE (Page Table Entry) mau pun TLB dianggap **valid**, jika isi **PTE (frame #)** bukan 0000. Bagan TLB (Translation Look-aside Buffer) sebagai berikut:

(PAGE #)	**(FRAME #)**
0000	0123
0010	9876

i) Sebutkan komponen PAGE#1 dari VA1: _____ (HEX)

j) Apakah PAGE#1 berada dalam TLB? YA / TIDAK (lingkari)

k) Apakah PAGE#1 berada dalam PT YA / TIDAK (lingkari)

l) Apakah ada nilai FRAME#1 yang valid pada (TLB atau PT)? YA / TIDAK (lingkari)

m) Jika YA, berapa nilai FRAME#1 _____ (HEX)

n) Jika YA, berapa PA1 _____ (HEX)

o) Sebutkan komponen PAGE#2 dari VA2: _____ (HEX)

p) Apakah PAGE#2 berada dalam TLB? YA / TIDAK (lingkari)

q) Apakah PAGE#2 berada dalam PT YA / TIDAK (lingkari)

r) Apakah ada nilai FRAME#2 yang valid pada (TLB atau PT)? YA / TIDAK (lingkari)

s) Jika YA, berapa nilai FRAME#2 (HEX) _____

t) Jika YA, berapa PA2 (HEX) _____

u) Sebutkan komponen PAGE#3 dari VA3: (HEX) _____

v) Apakah PAGE#3 berada dalam TLB? (lingkari) YA / TIDAK

w) Apakah PAGE#3 berada dalam PT (lingkari) YA / TIDAK

x) Apakah ada nilai FRAME#3 yang valid pada (TLB atau PT)? (lingkari) YA / TIDAK

y) Jika YA, berapa nilai FRAME#3 (HEX) _____

z) Jika YA, berapa PA3 (HEX) _____

aa) Masukan nilai BUNTUT dalam format int 32 bit / BIG ENDIAN ke PA1, PA2, PA3 (jika valid).

Address (HEX)	***+0***	***+1***	***+2***	***+3***	***+4***	***+5***	***+6***	***+7***	***+8***	***+9***	***+A***	***+B***	***+C***	***+D***	***+E***	***+F***

(2015-1) Memori

a) Tulis lengkap Nomor Pokok Mahasiswa (NPM) anda

b) Tulis angka paling kanan dari NPM anda

c) Konversi angka butir “b” menjadi “UNSIGNED 32-BITS HEX”

Tambahkan butir “c” dengan “HEX” berikut

0	**1**	**0**	**0**	**0**	**0**	**0**	**0**

d) Nilai butir “d” ini ialah sebuah alamat virtual 32 BITS (VA)

e) Perhatikan kembali nilai butir “b” di atas!

i. Jika (0 ≤ b ≤ 1), konversi “b” menjadi bilangan UNSIGNED 16 BITS HEX.

ii. Jika (2 ≤ b ≤ 3), konversi “b” menjadi bilangan UNSIGNED 24 BITS HEX.

iii. Jika (4 ≤ b ≤ 6), konversi “b” menjadi bilangan UNSIGNED 32 BITS HEX.

Tugas anda ialah menempatkan nilai “e” ke dalam alamat virtual “d” (32 BITS) dengan ukuran halaman (page) 4 kbyte. Silakan lanjutkan langkah-langkah berikut:

f) Tulis ulang nilai butir “d” di atas:

i. Komponen Page Table Index (Number) = ._____.

ii. Komponen Offset = ._____.

Ukuran PTE (Page Table Entry) dengan format BIG ENDIAN terdiri dari 4 digit HEX yaitu satu digit flag dan 3 digit FRAME NUMBER. PTE dianggap sah, jika nilai flag != 0. Alamat awal Page Table (atau PAGE TABLE INDEX #0) ialah **`00 1000 (HEX)`**. Berikut merupakan tabel memori fisik:

	+0	**+1**	**+2**	**+3**	**+4**	**+5**	**+6**	**+7**	**+8**	**+9**	**+A**	**+B**	**+C**	**+D**	**+E**	**+F**
00 1000	**71**	**00**	**71**	**01**	**71**	**02**	**71**	**03**	**71**	**04**	**71**	**05**	**71**	**06**	**71**	**07**
00 1010	**71**	**08**	**71**	**09**	**71**	**0A**	**71**	**0B**	**71**	**71**	**71**	**71**	**71**	**71**	**71**	**71**
...	...															

00 2000	72	00	72	01	72	02	72	03	72	04	72	05	72	06	72	07
...	...															
00 3000	73	00	73	01	73	02	73	03	73	04	73	05	73	06	73	07
...	...															
00 4000	74	00	74	01	74	02	74	03	74	04	74	05	74	06	74	07
...	...															

g) Jadi, alamat fisik (PA) dari alamat virtual di atas ialah ._____.
h) Silakan masukan nilai butir “e” ke alamat fisik tersebut.

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F

i) Sebutkan metoda penempatan yang dipilih: “LITTLE ENDIAN” atau “BIG ENDIAN”?

(2010-1) Page Replacement Algorithm

Diketahui spesifikasi sistem memori virtual sebuah proses sebagai berikut:

- *page replacement* menggunakan algoritma LRU (*Least Recently Used*).
- ukuran halaman (*page size*) adalah 200 bytes.
- jumlah frame yang tersedia sebanyak 3.
- proses akan mengakses alamat berturut-turut sebagai berikut:

```
823, 1112, 1223, 1444, 1777, 1555, 1606, 1899, 1500, 919
```

a) Tentukan *reference string* berdasarkan ukuran halaman tersebut di atas! (awal reference string dimulai dari 1, misalnya references string 1 = 0-199 byte)

b) Jika algoritma LRU diimplementasikan pada struktur data stack, isilah bagan stack dibawah ini:

Top of stack →

http://scele.cs.ui.ac.id/course/view.php?id=1362

c) Tentukan jumlah *page-fault* yang terjadi!
d) Berapa jumlah frame minimal yang harus diberikan agar jumlah page faultnya minimum?

(2011-2/UCB Spring 2000) Demand Paging

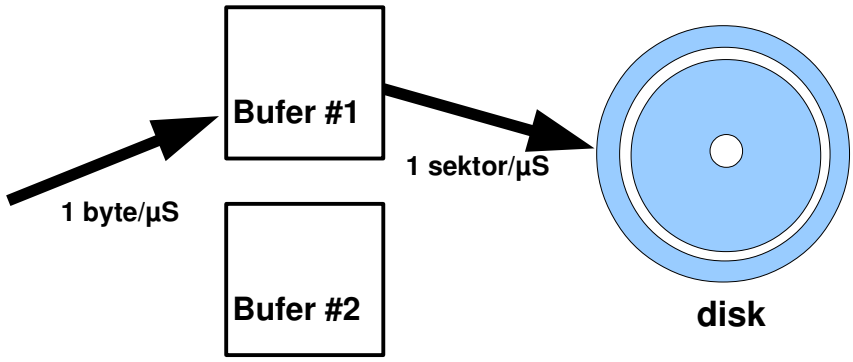
Perhatikan sistem "*demand paging*" dengan 4 (empat) bingkai/frame memori fisik untuk 7 (tujuh) halaman/*pages* dengan "*reference string*":

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6

Dengan asumsi bahwa bingkai memori pada awalnya kosong, berapa "page faults" akan terjadi, serta "halaman" mana saja yang tetap berada dalam bingkai memori fisik setelah *reference string* tersebut, jika menggunakan:

a) *FIFO page replacement policy*?
b) *LRU page replacement policy*?
c) *OPTIMAL page replacement policy*?

(2010-1) Disk



Diketahui sebuah disk dengan spesifikasi sebagai berikut:

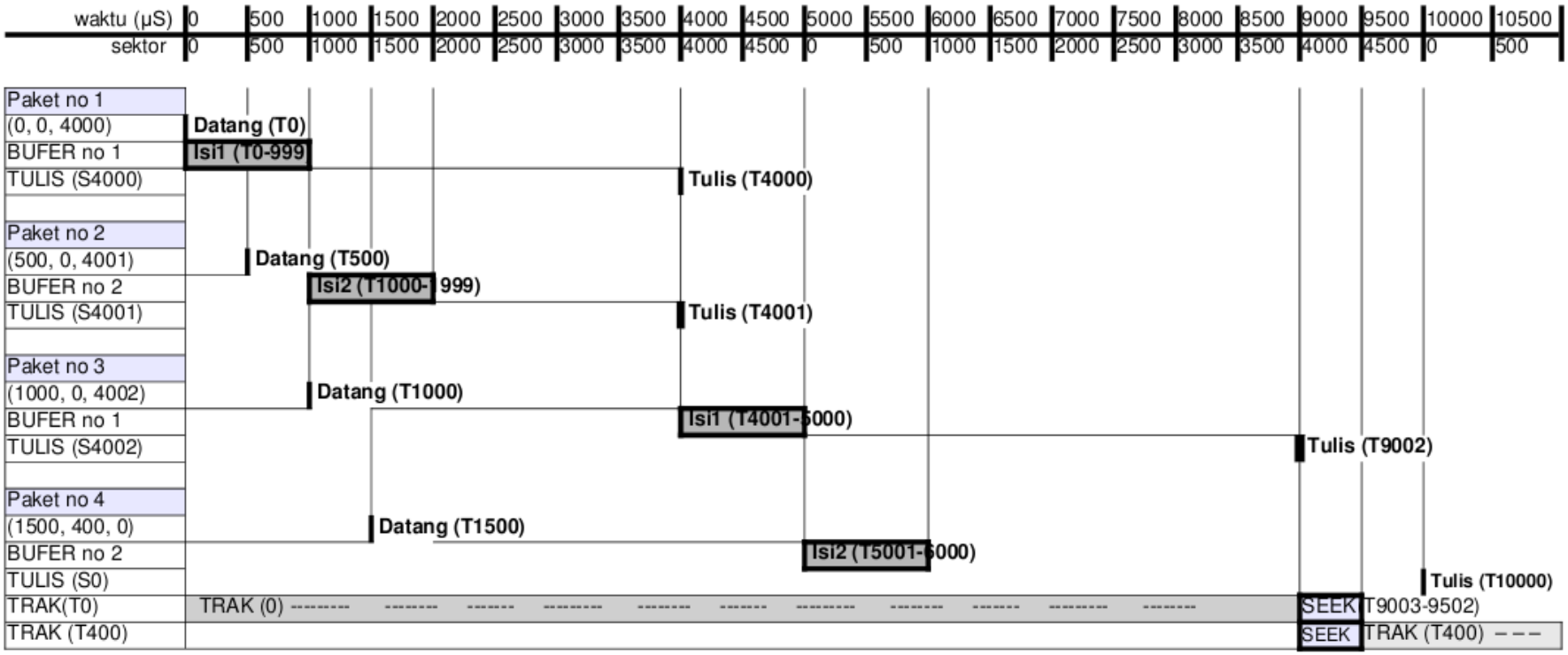
- 10000 silinder
- 5000 sektor per trak
- satu permukaan (*surface*) disk
- ukuran sektor = ukuran bufer = ukuran "paket" = 1 K-byte
- kecepatan menulis dari bufer ke sektor disk: 1 sektor per 1 µ-detik
- kecepatan menulis ke bufer dari sistem: 1 byte per 1 µ-detik
- waktu yang diperlukan sebuah *head* untuk pindah trak ("seek") ialah:
  - seek = (100 + Δ trak) µ-detik

  Umpama, untuk bergeser sebanyak 100 trak (Δ trak=100), *head* memerlukan waktu 100 + 100 = 200 µ-detik.
- anggap 1 G = 1000 M; 1 M = 1000 K ; 1 K = 1000 b
- pada saat t=0, head disk ada pada silinder=0, sektor=0
- pada satu saat, sistem operasi hanya dapat mengisi satu bufer
- sistem operasi hanya dapat mengisi bufer yang sudah kosong
- pada saat sistem operasi mengisi sebuah bufer, bufer lainnya secara bersamaan dapat menulis ke disk

a) Berapa kapasitas/ukuran disk?
b) Berapa RPM disk?
c) Diagram di halaman berikut merupakan contoh sistem dengan DUA BUFER yang melayani permintaan penulisan 4 paket ke disk. Tugas anda adalah membuat diagram serupa dengan sistem EMPAT BUFER yang melayani permintaan penulisan 4 paket yang sama.

Bagian III Disk

Nama/NPM _____



(0, 0, 4000)	artinya: pada t=0, ada permintaan tulis ke track=0, sektor=4000
TULIS (S4000)	artinya: tulis ke sektor=4000
TULIS (T4000)	artinya: tulis pada saat t=4000
Datang (T0)	artinya: permintaan tulis paket datang pada t=0
Isi1 (T0-999)	artinya: mengisi BUFER nomor 1 pada t=0 hingga dengan t=999
SEEK (T9003-9500)	artinya: seek dari t=9003 hingga t=9500

Bagian III Disk

Nama/NPM _____

waktu (μS)	0	500	1000	1500	2000	2500	3000	3500	4000	4500	5000	5500	6000	6500	7000	7500	8000	8500	9000	9500	10000	10500
sektor	0	500	1000	1500	2000	2500	3000	3500	4000	4500	0	500	1000	1500	2000	2500	3000	3500	4000	4500	0	500

Paket no 1
(0, 0, 4000)
Paket no 2
(500, 0, 4001)
Paket no 3
(1000, 0, 4002)
Paket no 4
(1500, 400, 0)

(2011-1) Disk

Diketahui sebuah hard disk (12000 RPM) dengan 1000 silinder yang masing-masing terdiri dari 21 permukaan. Setiap trak terdiri dari 1000 sektor dengan ukuran 1 kB. Dalam perhitungan ini, asumsikan 1 MB = 1000 KB.

a) Hitung kecepatan transfer data dari/ke sebuah trak.
b) Apa bila setiap peralihan silinder dibutuhkan 2 mS, sedangkan setiap peralihan trak dibutuhkan 10/101 mS; berapa total waktu yang dibutuhkan untuk transfer dari/ke disk data sebanyak 2121 MB?

(2011-2/Wikipedia) Disk IBM 350 Storage Unit

IBM 350 disk storage unit model 1 (151 cm x 171 cm x 50.2 cm) pertama kali diluncurkan pada tanggal 4 September 1956. IBM 350 memiliki 100 permukaan (*surfaces*) @ 100 *tracks*. Pada setiap trak ada 5 sektor @ 600 bit. Disk berputar sebanyak 1200 RPM. *Seek time* ialah (10 + 10 Tr) mS dimana “Tr” ialah jumlah pergeseran track. Abaikan “switching time” antar permukaan. Pada zamannya, IBM 350 dipergunakan untuk menyimpan karakter berbasis “6 bit”.

a) Ada berapa karakter berbasis "6 bit” dalam satu sektor?
b) Ada berapa karakter berbasis "6 bit” dalam satu trak?
c) Ada berapa karakter berbasis "6 bit” dalam satu silinder?
d) Ada berapa karakter berbasis "6 bit” dalam seluruh disk?
e) Berapakah kecepatan maksimum dari transfer data “teoritis” dalam satuan “karakter per detik”?
f) Berapa total waktu (mS) untuk menulis sektor demi sektor sebanyak 50100 karakter mulai (Trak 0, Permukaan 0, Sektor 0) atau (0, 0, 0) ke (0, 0, 4) ke (0, 1, 0) ke (0, 1, 4) ke (0, 2, 0) dan seterusnya.
g) (Trak, Permukaan, Sektor) mana yang akan ditulis paling akhir?

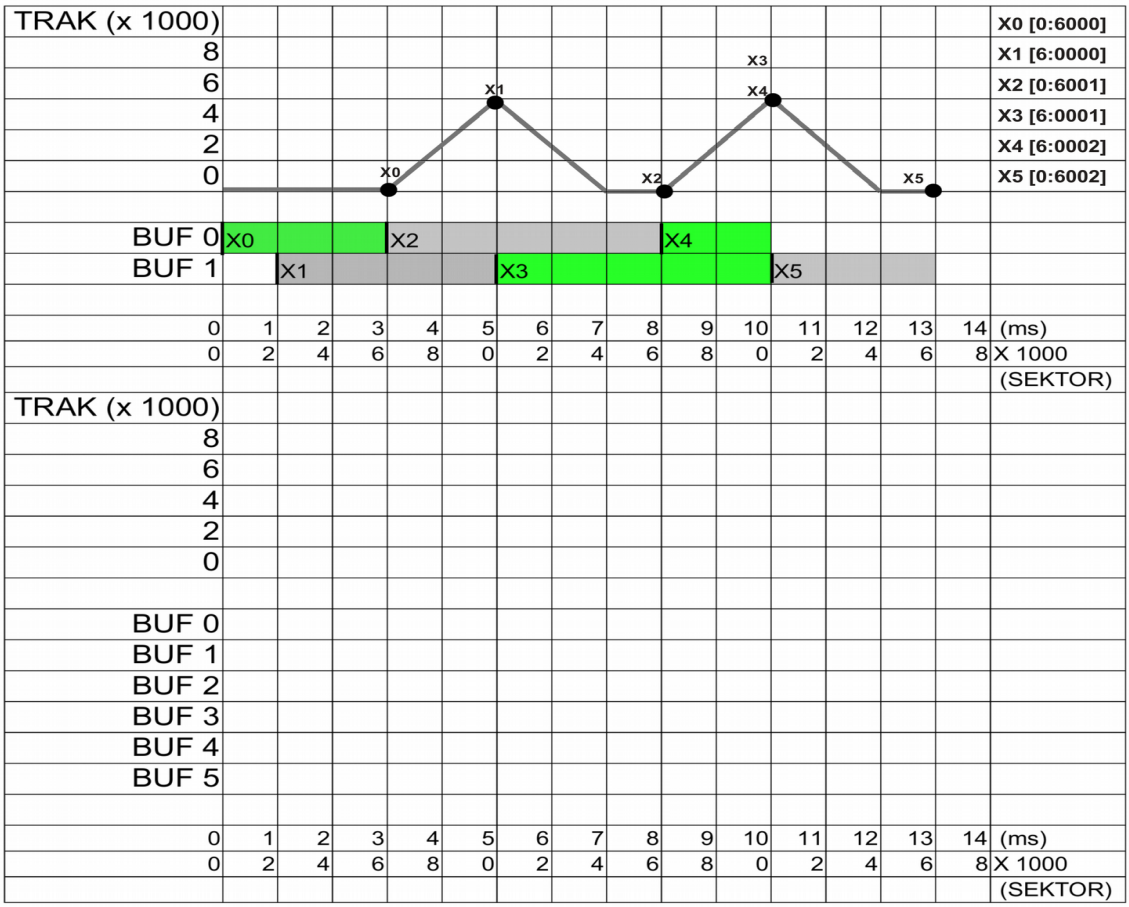
(2012-1) Disk

Diketahui sebuah disk permukaan tunggal (*single surface*) dengan 8000 silinder serta berputar 12000 rpm. Setiap trak terdiri dari 10000 sektor. Setiap sektor terdiri dari 10000 byte. Diasumsikan:

i. waktu untuk mengisi sebuah bufer mendekati nol.
ii. seek antar silinder (jauh/dekat) = 2 ms
iii. algoritma seek :
    1. Head tidak akan berpindah silinder selama ada buffer yang harus ditulis pada silinder tersebut
    2. Jika buffer telah kosong, maka penulisan antar silinder menggunakan algoritma FCFS (*First Come First Served*)

iv. 1T = 1 000G = 1 000 000M = 1 000 000 000k
v. Berikut contoh diagram penulisan disk dengan 2 buffer (BUF0 dan BUF1) berturut-turut X0 [0:0000:6000], X1 [1:6000:0000], X2 [2:0000:6001], X3 [3:6000:0001], X4 [4:6000:0002], X5 [5:0000:6002]; dimana [aaa:bbb:ccc] ialah aaa: waktu masuk bufer (t0=0) ms, bbb: trak, ccc: sektor.

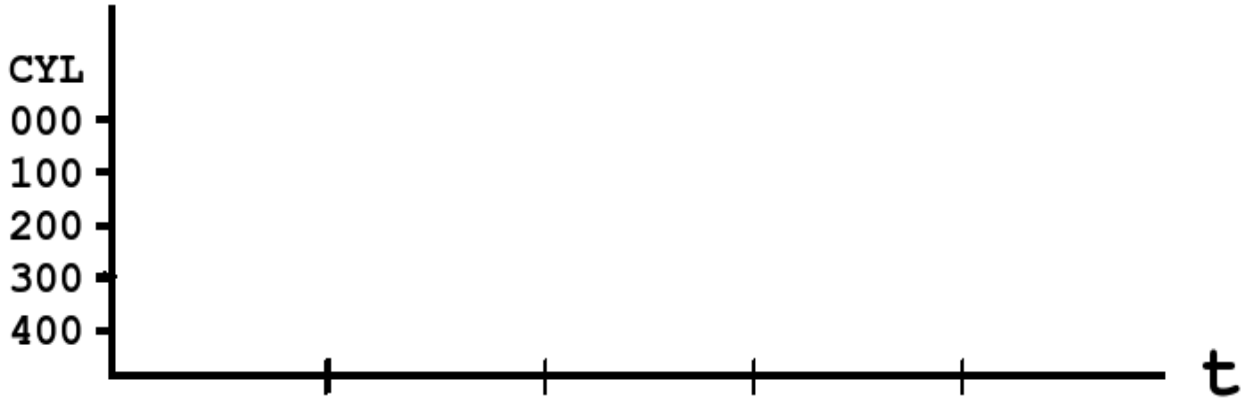
a) Berapa GB kapasitas disk?
b) Berapa GB/detik kecepatan transfer teoritis disk?
c) Buat diagram penulisan disk dengan 6 bufer (BUF0, BUF1, BUF2, BUF3, BUF4, BUF5).



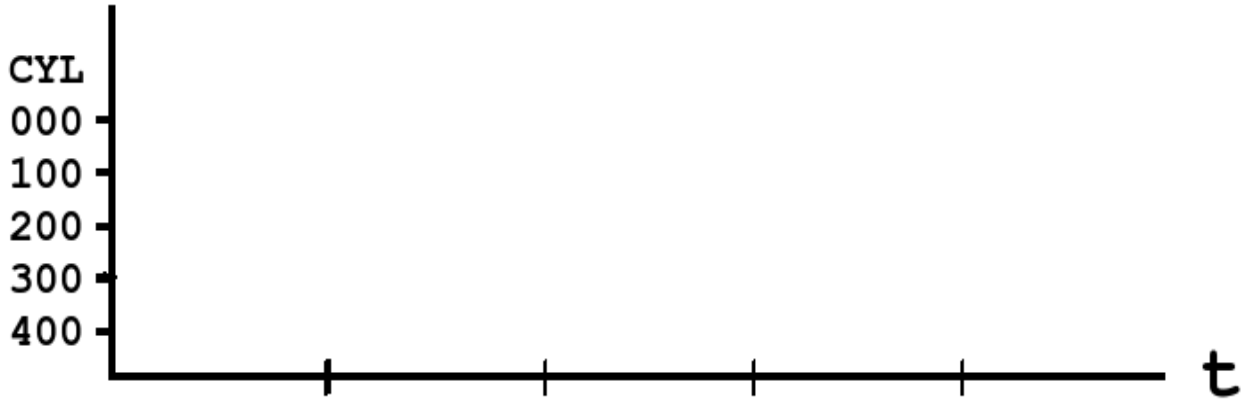
(2012-2) Disk

Diketahui HDD dengan ketentuan berikut: 6000 RPM, 25000 cylinders, 4 permukaan (surfaces), 10000 sectors, 1 k-byte per sector. Seek time = (800 + Δ track) µS. Gunakan 1 K = 1000; 1M = 1000K; 1G = 1000M.

a) Berapakah kapasitas total HDD dalam Giga-bytes?
b) Berapakah transfer rate maksimum HDD dalam Mega-bytes per detik?
c) Berapa lama (dalam µS) diperlukan untuk menulis/membaca sebuah sektor?



d) Saat t=0, posisi HDD (surface, cylinder, sector) ialah (0, 0, 0). Berapa waktu total yang diperlukan untuk menulis berturut-turut sektor-sektor berikut ini: (0, 0, 0) → (0, 0, 6000) → (0, 100, 4000) → (0, 200, 2000) → (0, 300, 8000)? Gambarkan dengan diagram.
e) Atur pemulisan sektor, agar waktu total penulisan menjadi minimum. Gambarkan dengan diagram.



(2013-1) Disk

Diketahui sebuah *Hard Disk Drive* (HDD) dengan ketentuan berikut: 12000 RPM, 50000 *cylinders*, satu permukaan (*surfaces*), 10000 sektor per trak, 10 kbyte per sektor. *Seek time* (flat) = 1 ms. Pada saat t=0, posisi head pada trak=0, sektor=0 atau (0,0). Gunakan 1 K = 1000; 1M = 1000K; 1G = 1000M; 1T = 1000G

a) Berapakah kapasitas total HDD dalam *Tera-bytes?*

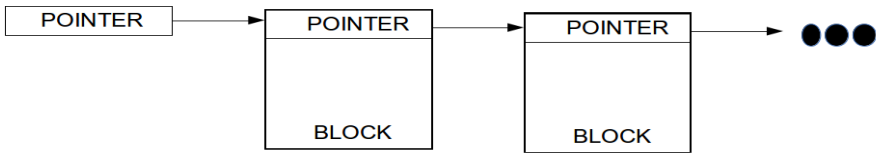
b) Berapakah transfer rate maksimum HDD dalam *Giga-bytes* per detik?

c) Berapa lama (dalam µS) diperlukan untuk menulis/membaca sebuah sektor?

d) Dari posisi (0, 0) tersebut di atas, berapa lama diperlukan untuk menulis sebanyak 10002 sektor dimulai dari trak=100, sektor=0 (100, 0) ke (100, 9999) lalu (101, 0) ke (101, 1)?

e) *Ulangi perhitungan "d)" di atas dengan ketentuan berbeda. Dari posisi (0, 0) tersebut di atas, penulisan tetap dimulai dari trak=100, namun tidak harus mulai dari sektor 0. Berapakah waktu* ***tersingkat*** *diperlukan untuk menulis 10002 sektor?*

(2013-2) Disk



Diketahui sebuah disk yang terdiri dari 256 M sektor. Rancang sebuah sistem berkas (*File System*) yang terbentuk dari *linked-list block*s (blok). Dalam setiap blok terdapat pointer yang menunjuk pada blok berikutnya. Ukuran pointer harus kelipatan 8 bit (1 byte). Ukuran satu blok = satu sektor = 8 KBytes. Gunakan asumsi, 1k=1024; 1M=1024k; 1G=1024M; 1T=1024G.

a) Berapa Tbyte ukuran disk?
b) Berapa bit, ukuran pointer yang akan digunakan?
c) Terangkan, mengapa ukuran maksimum berkas pasti akan kurang dari ukuran disk!

(2014-1) Sistem Berkas I-NODE dan DISK

Disk ***Seagate*** **SRD002** (7200 RPM) terdiri dari 255 permukaan lojik, 63 sektor/trak lojik, 243201 silinder lojik, dan total [3 907 029 167] sektor lojik. Ukuran sektor ialah 512 bytes. Gunakan sistem berkas berbasis **I-NODE** dengan 40 *direct pointers*, 4 *single indirect pointers*, 4 *double indirect pointer*, dan 4 *triple indirect pointer*. Ukuran pointer 32 bit.

a) Berapa jumlah kepala (***head***) lojik pada disk tersebut?

b) Berapa TeraByte kapasitas Disk tersebut (harap **dibulatkan** ke TeraByte terdekat!)?

c) Berapakah *transfer rate optimal* untuk sebuah trak (satuan: **sektor/detik**)?

d) Berapa ukuran berkas maksimum, jika hanya memanfaatkan **40 direct pointers**?

e) Berapa ukuran berkas maksimum, jika memanfaatkan ***40 direct pointers*** *dan* ***1 single indirect pointer?***

(2015-1) SISTEM BERKAS

FAT (***File Allocation Table***) merupakan sistem berkas yang dikembangkan pada tahun 1970an yang lalu. FAT-12 artinya *cluster* dengan *bit-space* 12 bit.

a) Hitung ukuran maksimum sistem berkas FAT-12, jika menggunakan *cluster* berukuran 8 kbytes.
b) Hitung juga, ukuran maksimum sistem berkas FAT-12, jika menggunakan *cluster* berukuran 64 kbyte.
c) Apa yang dapat dilakukan, jika kapasitas disk lebih besar dari maksimum ukuran sistem berkas yang ada?
d) Terangkan karakteristik fragmentasi internal terhadap ukuran *cluster* pada sistem berkas berbasis FAT-12!
e) Terangkan karakteristik fragmentasi eksternal pada sistem berkas berbasis FAT-12!

(2012-1) RAID

a) Konsep RAID 1 hingga RAID 5 diperkenalkan pertama kali oleh Patterson dkk. pada tahun 1988. Selanjutnya diperkenalkan konsep lainnya seperti RAID 0 (*stripping*), RAID 0 + RAID 1 (RAID 01), RAID 1 + RAID 0 (RAID 10). Apa bedanya RAID 01 dan RAID 10?
b) Selanjutnya muncul konsep RAID 6 yang mulai menggeser peranan RAID 5. Dimana letak keunggulan dan kerugian RAID 6 terhadap RAID 5?
c) Belakangan mulai diperkenalkan konsep bertingkat seperti RAID 60 (RAID 6 + RAID 0). Berapa jumlah disk minimum yang diperlukan untuk membuat RAID60?
d) Data Center Universitas Abal-Abal (**DC-UAA**) merencanakan sebuah SAN berbasis RAID60. Masing-masing disk yang akan digunakan berukuran 2 TB. Kapasitas DATA yang diinginkan setidaknya 20TB. Kecepatan akses yang diinginkan 3 x lebih cepat daripada menggunakan RAID6 biasa. Berapa jumlah minimum disk yang diperlukan?
e) Gambar diagram RAID DC-UAA seperti butir d tersebut di atas.
f) Apa yang harus dilakukan, agar Admin DC-UAA dapat mudah menambah disk data dikemudian hari? Pada saat itu, besar kemungkinan, ukuran standar satu disk bukan lagi 2 TB.

(2013-2) RAID

Diketahui sekumpulan disk masing-masing berukuran 1 Tbyte. Rancang sebuah kumpulan disk dengan kinerja sekurangnya 6 kali disk aslinya dengan jumlah disk seminimum mungkin.

a) Buat diagram untuk sistem tanpa toleransi kehandalan (*fault tolerance*) sama sekali! Berapa disk yang diperlukan?
b) Buat diagram untuk sistem RAID 6 + 0! Berapa disk yang diperlukan?

(2014-1) RAID

Sebuah sistem RAID 100 yang dibentuk dari kumpulan disk berukuran @ 1 Tera-byte.

a) Buat diagram RAID 100 tersebut dengan jumlah disk minimum.
b) Berapa disk yang diperlukan untuk konfigurasi minimum tersebut?
c) Berapa kapasitas DATA dari sistem tersebut?
d) Mana yang lebih cepat: MENULIS atau MEMBACA dari sistem RAID 100 tersebut? Terangkan!

(2015-1) RAID

Diketahui delapan (8) buah disk yang dapat menjadi berbagai susunan “array” seperti RAID-1, RAID-5, RAID-6, RAID-60, RAID-61, RAID-10, dan RAID-100. Dalam perhitungan di sini, gunakan **kecepatan** “baca atau tulis relatif” dibandingkan disk tunggal. Gunakan juga **kapasitas** “relatif” dibandingkan kapasitas sebuah disk. Sebutkan dan buatkan diagram dari susunan RAID mana tersebut di atas dengan:

a) kapasitas disk terbesar = _____ X untuk RAID _____

b) kecepatan membaca (relatif) tertinggi = _____ X untuk RAID _____

c) kecepatan menulis (relatif) tertinggi = _____ X untuk RAID _____

d) konfigurasi jumlah disk rusak terbanyak dengan tidak sampai kehilangan DATA = _____ DISK untuk RAID _____

e) konfigurasi jumlah disk rusak paling sedikit yang dapat menyebabkan kehilangan DATA = _____ DISK untuk RAID _____

(2011-1) Android-101

Android merupakan "*software stack*" untuk "*mobile devices*" yang terdiri dari lapisan/komponen-komponen utama seperti: *Linux Kernel, Native Libraries, Android Runtime, Application Framework,* dan *Applications*.

a) Sebutkan sekurangnya empat jenis "*mobile devices*" yang mendukung Android.
b) Sebutkan sekurangnya empat *Android Standard System Applications*.
c) Apa yang dimaksud dengan Linux Kernel sebagai "*Hardware Abstraction Layer*". Sebutkan juga keuntungan penggunaan Linux Kernel dalam Android!
d) Bagaimana caranya Android Application memanfaatkan Kernel Linux?
e) Sebutkan keunggulan Dalvik VM dibandingkan VM lainnya.

(2011-1) Sistem Hitung Gaji/Lembur JADUL

Sebuah Sistem Hitung Gaji/Lembur dioperasikan setiap akhir bulan selama 100 jam non-stop. Sistem JADUL ini -- digunakan sejak tahun 1998 -- berbasis teknologi CPU 1500 MHz serta Hard Disk 600 RPM dengan waktu seek rata-rata 20 mS.
Berdasarkan analisa konsultan dari PUSILKOM UI, 80% dari waktu sistem dipergunakan untuk operasi I/O, sedangkan sisanya merupakan operasi proses/CPU. Analisa yang lebih dalam menunjukkan bahwa 80% dari operasi I/O berlangsung di Hard Disk dengan perbandingan antara waktu "*seek*" dan "*rotational latency*" yaitu 50%:50%.

a) Untuk meningkatkan kinerja, CPU diganti dengan yang arsitektur serupa namum lebih cepat yaitu 3GHz. Hitung, waktu yang dibutuhkan untuk menghitung Gaji/Lembur dengan CPU baru tersebut.
b) Karena peningkatan kinerja dianggap kurang memuaskan, Hard Disk diganti dengan yang lebih cepat yaitu 4800 RPM dan seek rata-rata 5 mS. Hitung, waktu yang dibutuhkan untuk menghitung Gaji/Lembur dengan CPU dan Hard Disk baru tersebut.
c) Jika proses dijalankan pada akhir bulan, tanggal 28 Februari 2012 jam 08:00 pagi; kapan proses akan rampung? Apakah dapat rampung sebelum 1 Maret 2012 jam 0:00?

(2012-2) Kinerja

Diketahui sebuah aplikasi dengan waktu total eksekusi 100 jam. Berdasarkan analisa, 24% dari waktu eksekusi ialah waktu CPU, sedangkan sisanya ialah waktu menangani HDD. Umpamanya, waktu total diinginkan kurang dari 50 jam. Pertama, CPU diganti dengan yang memiliki 4 (empat) core yang masing-masing dua kali lebih cepat dari sebelumnya. Kedua, mempersiapkan sebuah sistem RAID 6+0 menggantian sistem lama, namun dengan kinerja HDD yang sama.

a) Gambarkan diagram RAID 6+0 yang terkait!
b) Berapa kapasistas minimum dari konfigurasi HDD yang baru?

(2013-1) Kinerja

Diketahui sebuah Aplikasi dengan waktu total eksekusi 100 jam. Berdasarkan analisa, 16% dari waktu eksekusi ialah waktu CPU, 60% dari waktu eksekusi merupakan operasi Hard Disk, serta sisanya (24%) merupakan operasi lainnya. Untuk meningkatkan kinerja, yang semula “CPU *core* tunggal” diganti dengan yang “CPU 4 *core*”. Masing-masing “*core*” lebih cepat dua kali dibandingkan CPU aslinya.

Kini, tugas anda ialah merancang sistem RAID 6+0, agar waktu total eksekusi sistem yang baru dibawah 32 jam! Jadi, waktu eksekusi sistem RAID 6+0 harus 10 (sepuluh) kali lebih cepat dari sistem disk aslinya.

a) Rancang susunan RAID 6+0 yang diperlukan
b) Berapa jumlah disk (minimum) yang diperlukan sistem RAID 6+0 ini?
c) Berapa peningkatan (minimum) kapasitas penyimpanan dibandingkan sistem semula?

Revision: 716 -- 15 Nov 2017 URL: http://rms46.vLSM.org/2/183.pdf