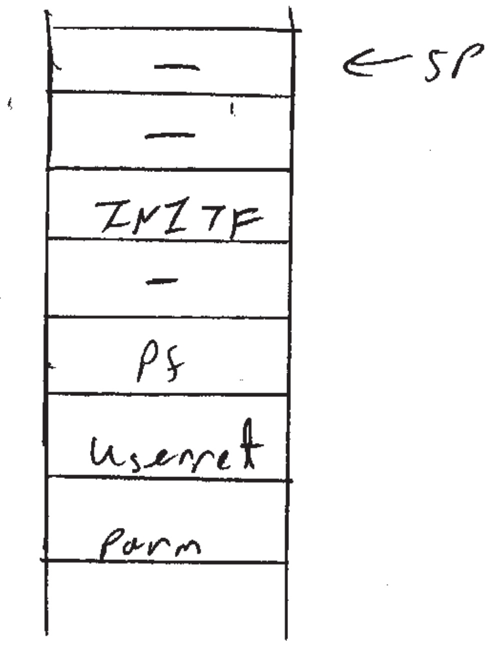
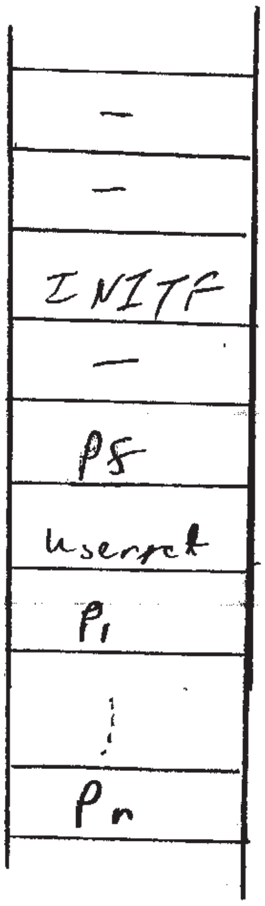
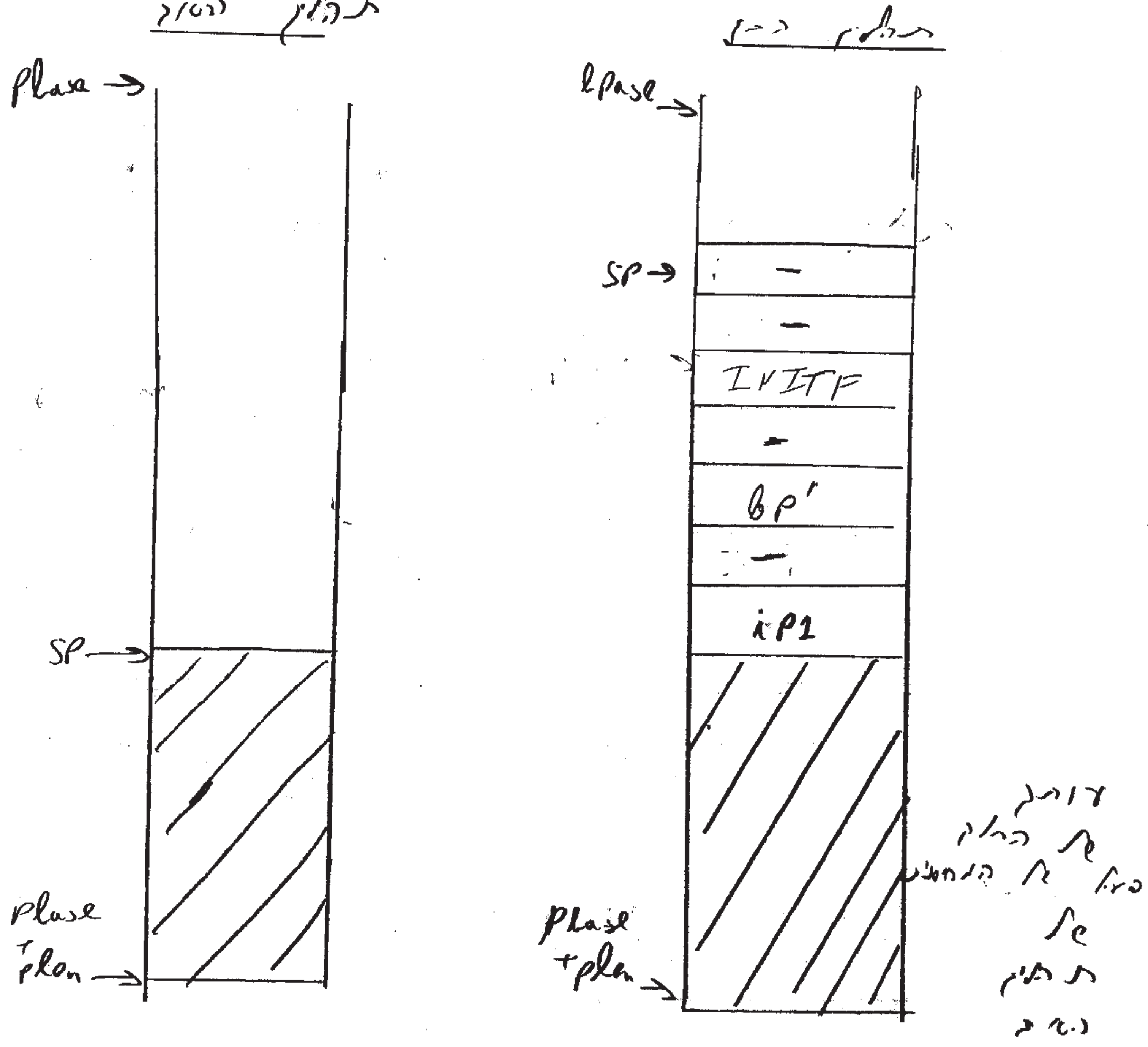
המחסנית של xexecl



המחסנית של xexecl2



המחסניות של xfork



```
 /* xexecl.c - xmain, prA, prB */

#include <conf.h>
#include <kernel.h>
#include <io.h>
#include <proc.h>
#include <sem.h>
#include <mem.h>
#include <q.h>
#include <bios.h>
#include <kbdio.h>

#define INITF 0x0200

extern int INITRET();

/*------------------------------------------------------------------------
 *  xmain  --  example of creating processes in PC-Xinu
 *------------------------------------------------------------------------
 */

void prA(), prB();

xmain()
{

        resume( create(prA, INITSTK, INITPRIO, "proc 1", 1,  'A') );
}

/* xexecl - emulate unix execl in xinu */

xexecl(void (*pf)(), int parm)
{
struct pentry *pptr;
char *saddr;
int *sp1;
int ps;
int dummy;

disable(ps);
pptr = &proctab[currpid];
pptr->phasmsg = 0;
sp1 = (int *)(pptr->pbase + pptr->plen);
pptr->pargs = 1;
*(--sp1) = parm;
*(--sp1) = (int) INITRET;
*(--sp1)= (int) pf;
--sp1;
*(--sp1) = INITF;
sp1 -= 2;
pptr->pregs = sp1;
pptr->paddr = pf;

ctxsw(&dummy, &pptr->pregs);

 } /* xexecl */
```

```
/*------------------------------------------------------------------------
 *  prA  --  repeatedly print 'A' without ever terminating
 *------------------------------------------------------------------------
 */

void prA(int ch)
 {
   int i;
         for(i=0; i< 12; i++)
          {
                 putc(CONSOLE, ch);
                 putc(CONSOLE, '\n');
          }
         sleep(5);
         xexecl(prB, 'B');
 }

 /*------------------------------------------------------------------------
  *  prB  --  repeatedly print 'B' without ever terminating
  *------------------------------------------------------------------------
  */
void prB(int ch)
{
  int i;
        for(i=0; i< 12*80; i++)
                putc(CONSOLE, ch);
}
```

```
A
A
A
A
A
A
A
A
A
A
A
A
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
^@

-- system halt --

PC-Xinu terminated with 4 processes active
Returning . . .
```

```
/* xexecl2.c - xmain, prA, prB */

#include <conf.h>
#include <kernel.h>
#include <io.h>
#include <proc.h>
#include <sem.h>
#include <mem.h>
#include <q.h>
#include <bios.h>
#include <kbdio.h>

#define INITF 0x0200

extern int INITRET();

/*------------------------------------------------------------------------
 *  xmain  --  example of creating processes in PC-Xinu
 *------------------------------------------------------------------------
 */

void prA(), prch123();

xmain()
{

        resume( create(prA, INITSTK, INITPRIO, "proc 1", 1,  'A') );
}

/* xexecl2 - emulate unix execl in xinu */

xexecl2(void (*pf)(), int n, ...)
{
struct pentry *pptr;
char *saddr;
int *sp1, *a;
int ps, i;
int dummy;

disable(ps);
pptr = &proctab[currpid];
pptr->phasmsg = 0;
sp1 = (int *)(pptr->pbase + pptr->plen);

pptr->pargs = n;
a = n + 1 + &n;
for(i=0; i < n; i++)
  *(--sp1) = *(--a);

*(--sp1) = (int) INITRET;
*(--sp1)= (int) pf;
--sp1;
*(--sp1) = INITF;
sp1 -= 2;
pptr->pregs = sp1;
pptr->paddr = pf;

ctxsw(&dummy, &pptr->pregs);

} /* xexecl2 */
```

```
/*------------------------------------------------------------------------
 *  prA  --  repeatedly print 'A' without ever terminating
 *------------------------------------------------------------------------
 */

void prA(int ch)
{
  int i;
        for(i=0; i< 10; i++)
         {
                putc(CONSOLE, ch);
                putc(CONSOLE, '\n');
         }
        sleep(5);
        xexecl2(prch123, 3, 'B', 'C', 'D');
}

/*------------------------------------------------------------------------
 *  prch123  --  repeatedly print 'ch1ch2ch3' without ever terminating
 *------------------------------------------------------------------------
 */
void prch123(int ch1, int ch2, int ch3)
{
  int i;
        for(i=0; i< 10; i++)
         {
                putc(CONSOLE, ch1);
                putc(CONSOLE, ch2);
                putc(CONSOLE, ch3);
                putc(CONSOLE, '\n');
         } /* for */

} /* prch123 */
```

```
A
A
A
A
A
A
A
A
A
A
BCD
BCD
BCD
BCD
BCD
BCD
BCD
BCD
BCD
BCD
```

```
 /* xfork.c - xmain, prA, prB */

#include <conf.h>
#include <kernel.h>
#include <io.h>
#include <proc.h>
#include <sem.h>
#include <mem.h>
#include <q.h>
#include <bios.h>
#include <kbdio.h>

#define INITF 0x0200

extern int INITRET();


/* retip - compute ip of point of program */

int retip()
{
int ip1;

  asm {
        push ax
        mov ax,[BP+2]
        mov ip1,ax
        pop ax
      }
  return ip1;
}

/* xfork - xinu emulation of unix fork, will work
   only in the process main program, and pointers should not be used -
   pointers in the child process will point into the parent variable
   space */

int xfork()
{
char *saddr;
int *sp1, *sp2, *sp3, *sp4;
int ps, bp1;
int dummy;
int pid;
struct pentry *pptr, *pptr1;
int ip1;


disable(ps);
pptr = &proctab[currpid];
pid = create(pptr->paddr,pptr->plen, pptr->pprio, pptr->pname,0);

if (pid == SYSERR)
{
  restore(ps);
  return SYSERR;
} /* if */

pptr1 = &proctab[pid];

asm mov sp1,sp
sp2 = pptr->pbase + pptr->plen;
sp3 = pptr1->pbase + pptr1->plen;
/* give child process a duplicate stack */

for(;sp2 >= sp1;)
{
  *sp3 = *sp2;
  sp2--;
  sp3--;
}

/* compute instruction pointer for child process */

ip1 = retip();

/* child process  starts HERE */

if (currpid != pid) /* parent process only */
   {
        *(int *)sp3 = ip1;      /* simulate a context switch    */
        sp3 -= 1;

        /* simulate call to ctxsw */

        /* bp adjusting - necessary because our xinu does not support
              virtual addressing, but rather uses real addressing */

        /* bp adjusting of ctxsw for child process - real mode */

        asm mov bp1,bp
        *(int *)sp3 = ((int)pptr1->pbase) + ((bp1 -((int)pptr->pbase)));
        sp3 -= 1;                                  /* 1 word for bp              */
        *(int *)sp3 = INITF;             /* FLAGS value                */
        sp3 -= 1;
        sp3 -= 1;                        /* 2 words for si and di      */

        /* complete emulation of ctxsw */

        pptr1->pregs = sp3;

        /* bp adjusting of xfork for child process - real mode */
        asm mov bp1,bp
        sp4 =(int *) ( ((int)pptr1->pbase) + (( bp1 -((int)pptr->pbase)) ));

        /* bp adjusting of xmain for child process - real mode */
        asm {
              push ax
              mov ax,[bp]
              mov bp1,ax
              pop ax
             }
         *sp4 = ( ((int)pptr1->pbase) + (( bp1 -((int)pptr->pbase)) ));

     resume(pid);
     restore(ps);
     return pid;


    } /* if */
  else
   return 0;   /* child process only */

 } /* xfork */
```

```c
 /* tstxfrk.c - test xfork */

#include <conf.h>
#include <kernel.h>
#include <io.h>
#include <proc.h>
#include <sem.h>
#include <mem.h>
#include <q.h>
#include <bios.h>
#include <kbdio.h>



/*------------------------------------------------------------------------
 *  xmain  --  example of creating processes in PC-Xinu
 *------------------------------------------------------------------------
 */

void process()
{
  int n = 100;

  int id, *nptr;

  nptr = &n;
  if ( ( id =  xfork() ) == 0 )
  {  /* select child process */
     printf("\n**************  child process ***********\n");

     *nptr = 999;  /* Only this line is different */

     printf("PID is %d and ID is %d.\n", getpid(), id);
     printf("n is %d, *nptr is %d and nptr is %d.\n", n, *nptr, nptr);
     printf("\n**************  child process ***********\n");

     sleep(6);
     printf("\n Press enter to continue ... ");
     getchar();

     printf("\n**************  child process ***********\n");
     printf("PID is %d and ID is %d.\n", getpid(), id);
     printf("n is %d, *nptr is %d and nptr is %d.\n", n, *nptr, nptr);
     printf("\n**************  child process ***********\n");

     n = 707;

     printf("\n**************  child process ***********\n");
     printf("PID is %d and ID is %d.\n", getpid(), id);
     printf("n is %d, *nptr is %d and nptr is %d.\n", n, *nptr, nptr);
     printf("\n**************  child process ***********\n");

     return;
   }
```

```
     sleep(5);
     printf("\n**************  parent process ***********\n");
     printf("PID is %d and ID is %d.\n", getpid(), id);
     printf("n is %d, *nptr is %d and nptr is %d.\n", n, *nptr, nptr);
     printf("\n**************  parent process ***********\n");

     n = 200;
     *nptr = 300;
     printf("\n**************  parent process ***********\n");
     printf("PID is %d and ID is %d.\n", getpid(), id);
     printf("n is %d, *nptr is %d and nptr is %d.\n", n, *nptr, nptr);
     printf("\n**************  parent process ***********\n");
     while(1)
       ;

} /* process */

xmain()
{
resume(create(process, INITSTK, INITPRIO, "process",0));
} /* xmain */
```

```
E:\USERS\EYTAN\XINU4WIN\NEWSRC\EXAMPLES>tstxfrk
Initializing . . .

PC-Xinu Version 6pc (1-Dec-87)
63864 real mem
18312 base addr
45552 avail mem

Hit any key to continue . . .

**************  child process ***********
PID is 24 and ID is 0.
n is 100, *nptr is 999 and nptr is 26138.

**************  child process ***********

**************  parent process ***********
PID is 25 and ID is 24.
n is 999, *nptr is 999 and nptr is 26138.

**************  parent process ***********

**************  parent process ***********
PID is 25 and ID is 24.
n is 300, *nptr is 300 and nptr is 26138.

**************  parent process ***********

 Press enter to continue ...

**************  child process ***********
PID is 24 and ID is 0.
n is 100, *nptr is 300 and nptr is 26138.

**************  child process ***********

**************  child process ***********
PID is 24 and ID is 0.
n is 707, *nptr is 300 and nptr is 26138.

**************  child process ***********
```