

1. 2016-1 (McGill Fall 1998)

Asumsikan:

- i. Arsitektur komputer dengan ukuran halaman (page size) 1024 bytes.
- ii. Setiap karakter (char) menempati 1 alamat memori @ 1 byte.
- iii. Struktur data "matrix (baris,kolom)" untuk selanjutnya disebut "matrix".
- iv. Setiap 4 baris (berurutan) "matrix" berada dalam satu halaman (page).
- v. Setiap saat, maksimum ada 1 halaman (page) "matrix" dalam memori.
- vi. Saat awal eksekusi fungsi, tidak ada halaman (page) "matrix" dalam memori.

Lingkari atau beri silang huruf "B" jika betul, dan "S" jika salah.

B / S Fragmentasi eksternal (external fragmentation) akan terjadi pada sistem berbasis halaman (paging systems).

B / S Bingkai (frame) pada memori virtual (VM) dipetakan ke halaman (page) pada memori fisik.

B / S Pengeksekusian program berbasis demand paging selalu menghasilkan page fault.

B / S Sebuah fungsi, mungkin saja menempati lebih dari satu halaman (page).

```
011 void isiMatrix1 (){
012     char matrix[256][256];
013     int ii, jj;
014     for (ii=0; ii<8; ii++) {
015         for (jj=0; jj<256; jj++) {
016             matrix[ii, jj] = 'x';
017         }
018     }
019 }
```

B / S Setiap eksekusi baris 016, selalu akan terjadi "page fault" pada matrix.

B / S Pada seluruh iterasi loop luar baris 014-018, akan terjadi 8 kali "page fault" pada matrix.

B / S Pada saat mengeksekusi fungsi isiMatrix1(), terdapat kemungkinan terjadi TOTAL¹ lebih dari 3 kali "page fault".

```
021 void isiMatrix2 (){
022     char matrix[256][256];
023     int ii, jj;
024     for (jj=0; jj<256; jj++) {
025         for (ii=0; ii<8; ii++) {
026             matrix[ii, jj] = 'x';
027         }
028     }
029 }
```

B / S Terdapat kemungkinan terjadi TOTAL 2 kali "page fault" saat mengeksekusi baris 026.

B / S Pada setiap iterasi loop dalam baris 025-027, terjadi 2 kali "page fault" pada matrix.

B / S Pada seluruh iterasi loop luar baris 024-028, terjadi 512 kali "page fault" pada matrix.

2. 2016-2 (Waterloo 2012)

Page Table.

Consider this following "structure addressspace" of a 32-bit processor.

```
struct addressspace {
    vaddr_t as_vbase1      = 0x00100000; /* text segment: virtual base addr */
    paddr_t as_pbase1      = 0x10000000; /* text segment: physical base addr */
    size_t  as_npages1     = 0x20;      /* text segment: number of pages */
    vaddr_t as_vbase2      = 0x00200000; /* data segment: virtual base addr */
    paddr_t as_pbase2      = 0x20000000; /* data segment: physical base addr */
    size_t  as_npages2     = 0x20;      /* data segment: number of pages */
    vaddr_t as_vbase3      = 0x80000000; /* stack segment: virtual base addr */
    paddr_t as_pbase3      = 0x80000000; /* stack segment: physical base addr */
    size_t  as_npages3     = 0x10;      /* data segment: number of pages */
    int     page_size      = 0x1000;    /* virtual page size is 0x1000 bytes */
};
```

When possible, translate the provided address.

Possible	Virtual Address	Physical Address	Segment
YES	0x0010 0000	0x1000 0000	text
NO	0x0030 0000	—	—
	0x0010 FEDC		
	0x0011 0000		
	0x7FFF FFFF		
		0x2000 1234	
		0x8000 FFFF	