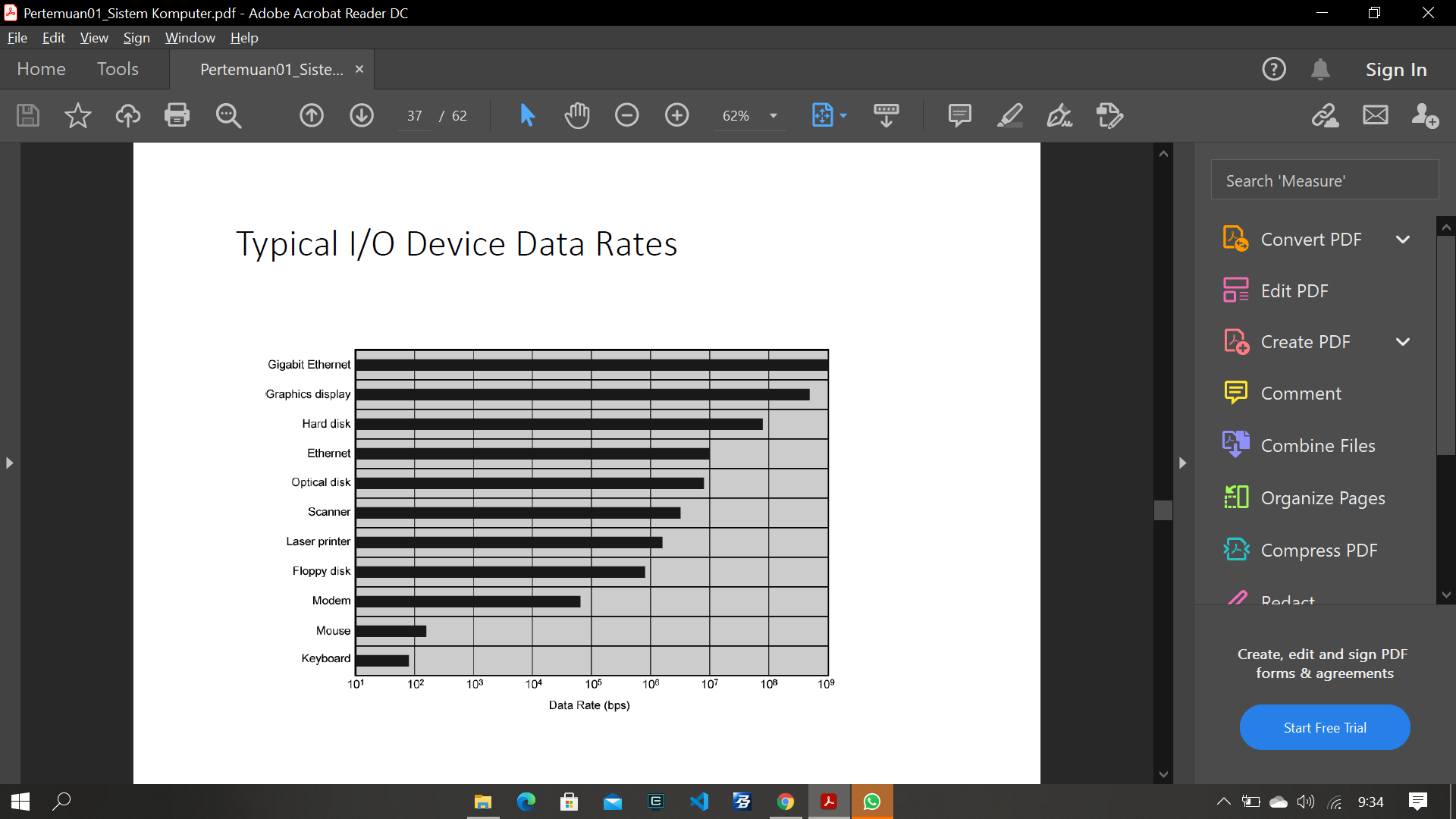
Kecepatan I/O device yang ada di pc/laptop :

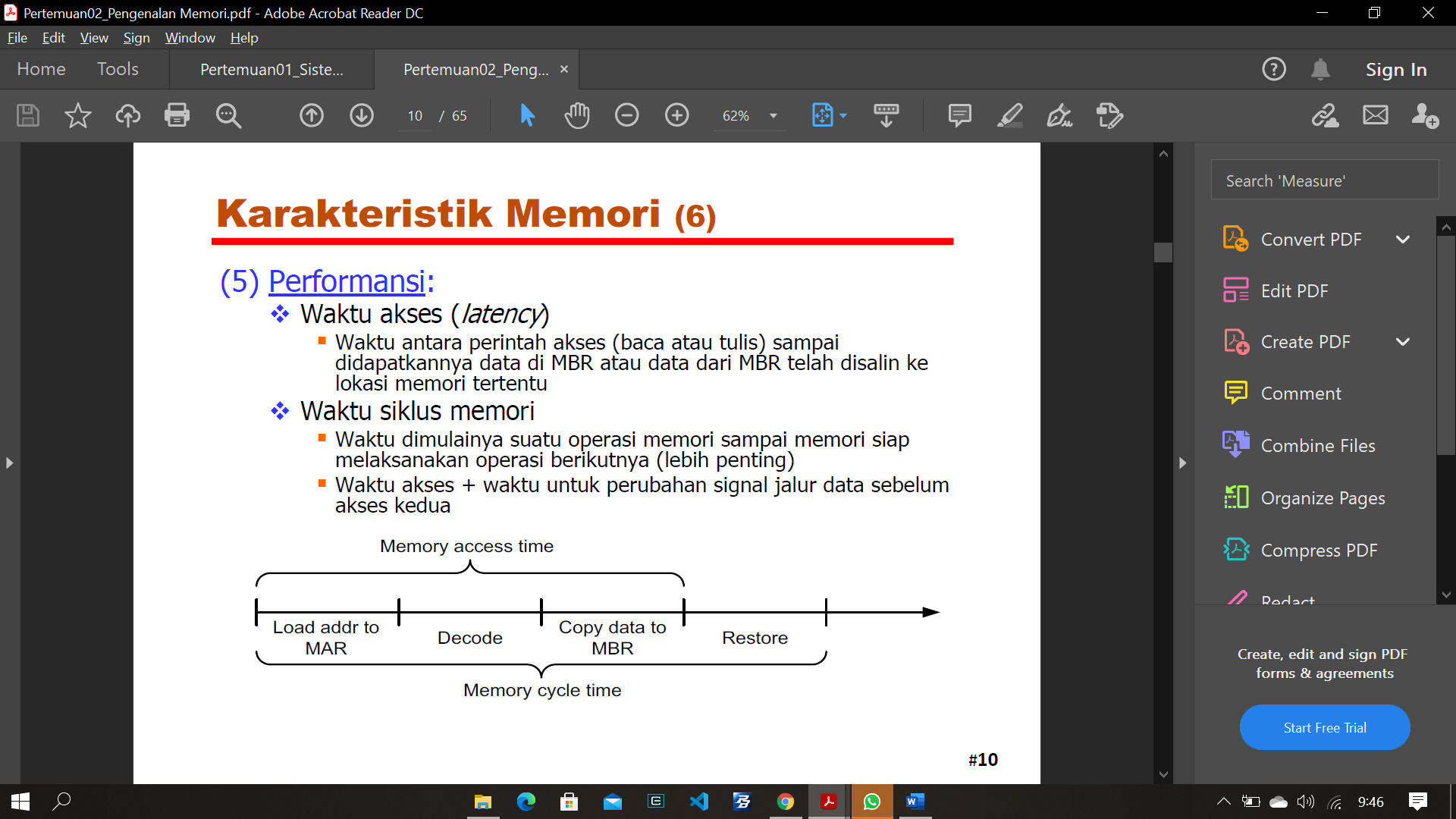


Memori :

1. Lokasi : Internal (RAM, Cache memory) (tanpa harus ada I/O)

Eksternal (Harddisk, flash disk) (harus ada I/O)

1. Kapasitas : Kemampuan menampung data dalam satuan tertentu (Byte/word), satu byte = 8bit, satu word = 8, 16, atau 32 bit
2. Bit Rate : banyak bit yang dapat di baca atau tulis dalam satu detik, memori internal berbanding lurus dengan lebar bus, memori eksternal digunakan block yang ukurannya lebih dari satu word, satuan alamat bergantung pembuat prosesor
3. Cara akses : sequential (secara berurutan, menggunakan shared read/write, waktu akses bervariable, contoh magnentic tape), direct (akses ke memori terdekat, memiliki alamat unik, waktu akses bervariable, contoh harddisk), random access (random langsung ke alamat memori, mempunyai alamat unik, waktu aksesnya konstan, contoh main memori), associative (dilakukan dengan membandingkan seluruh word secara bersamaan, waktu konstan, contoh cache memori)
4. Performansi : waktu akses ( lamanya akses sampai mendapatkan MBR), waktu siklus memori (waktu dimulainya suatu operasi sampai memori siap, waktu akses +waktu untuk perubahan signal jalur.



Transfer rate (waktu rata rata perpindahan data, rumus : Tn=Ta+N/R (Tn = waktu rata rata,Ta = rata rata tulis akses, N = jumlah bit, R = transfer rate)

1. Jenis fisik : semi konduktor (RAM, Flashdisk) Magentik (harddisk, magnetic tape), optic (CD,DVD)
2. Karakteristik fisik : volatile (nilainya hilang bila tidak ada listrik), non volatile (data ada walau tidak ada listik), erasable (nilainya dapat di hapus), non-erasable (tidak dapat dihapus nilainya)
3. Organisasi memori : penyusun bit untuk word.

RAM (Random Access Memory)

DRAM (Dynamic)

Data berupa muatan listrik di kapasitor, dan harus selalu di refresh. (main memory)

+ sederhana, kecil, murah, kapasitas besar,

* Perlu refresh, biaya refresh lebih mahal, lebih lambat

SRAM(Static)

Disusun dari beberapa transistor, tidak harus selalu di refresh. (cache memory)

+ tidak perlu refresh, tidak perlu rangkain refresh, cepat

* Komplek, besar, mahal, kapasitas kecil.

Advanca DRAM = SDRAM, RDRAM

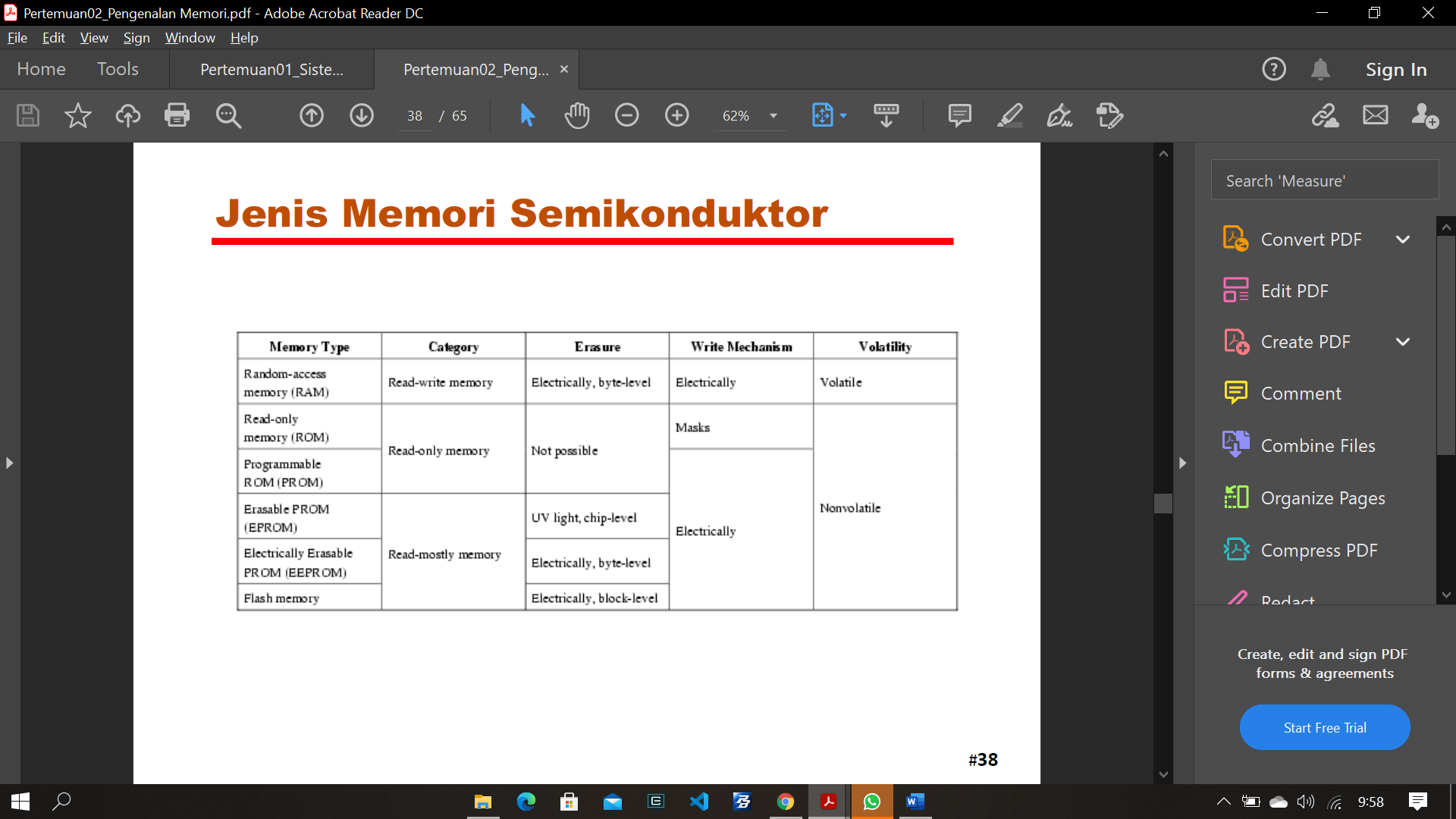
ROM (Read Only Memory) = data tidak dapat dihapus, bersifat permanen, ditulis saat membuat chip. Aplikasi : store firmware, BIOS

PROM =Programmable =dituli sesudah chip dibuat, hanya bisa satu kali tulis,non volatile, data tidak bisa dihapus. Aplikasi :RFID

EPROM =Erasable = dihapus menggunakan ultra violate tetapi data satu chip terhapus

EEPROM = Electically = dihapus secara terus menerus karena dengan listrik

Flas Memory = dihapus secara satuan blok memori.

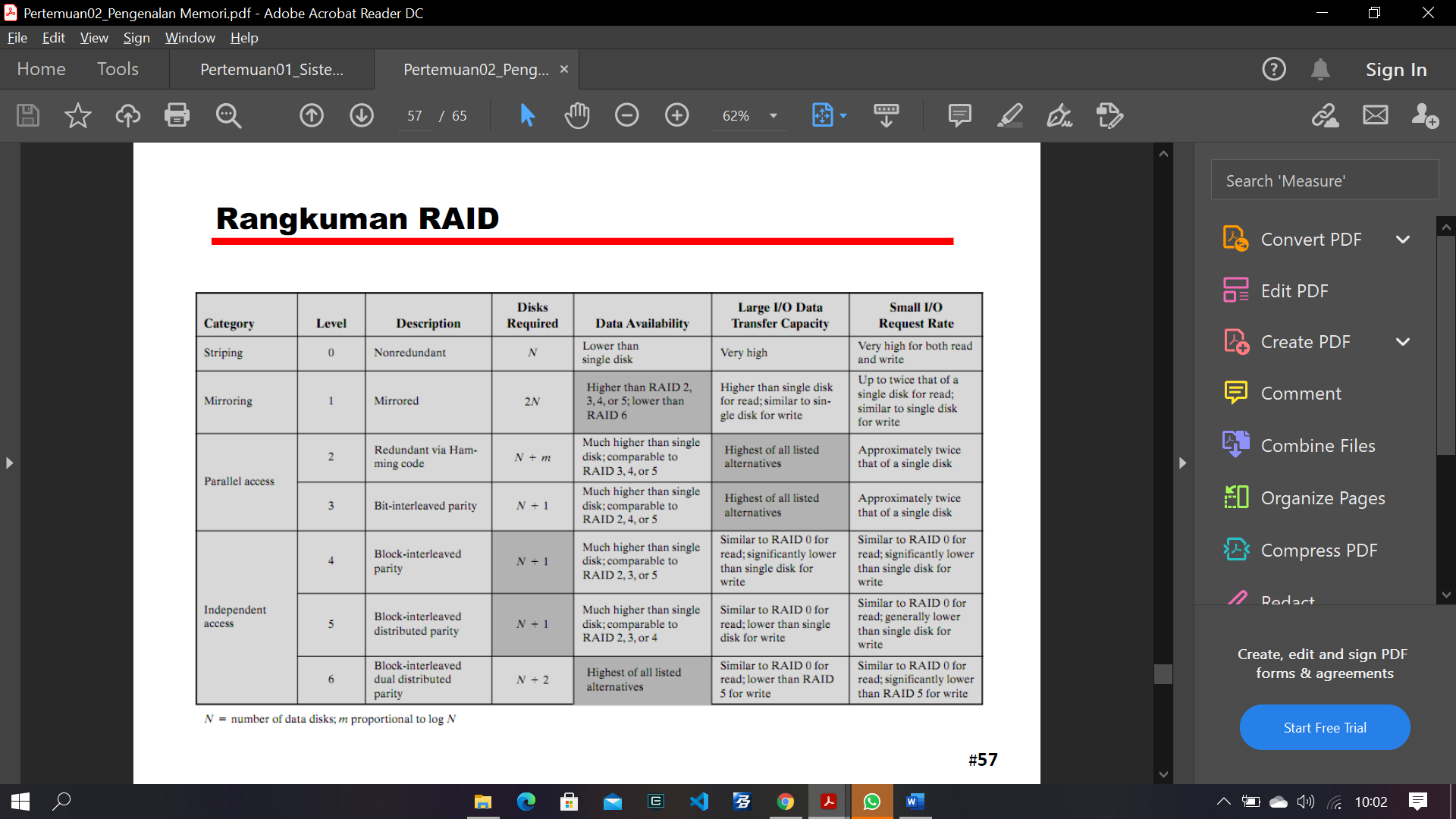


Error check = parity & hamming code

Parity = odd (ganjil jika 1nya genap maka tambah 1 agar ganjil), even (jika 1nya ganjil maka tambah 1 agar genap)

Hamming code seperti yang pernah dijelaskan (Panjang pisan)

Memori sekunder (RAID & Optik)

RAID = redundant array of independent/inexpensive disk.

Memori Optik = CD ROM, DVD R/RW

CD ROM = disimpan dalam bentuk pits, cara baca menggunakan pantulan laser, kerapatan dan kecepatan konstan, CD-R =hanya bisa diisi data sekali, CD-RW = bisa diisi data berkali kali.

DVD = video/versatile = data video/komputer.

DVD-5: satu sisi dan satu lapis, kapasitas total = 4,37 GB

DVD-9: satu sisi dan dua lapis dimana kapasitas setiap lapisan

adalah 4,37 GB dan 7,95 GB, sehingga kapasitas total

menjadi 12,32 GB

DVD-10: dua sisi masing-masing satu lapis, kapasitas total

sebesar 8,74 GB

DVD-18: dua sisi masing-masing dua lapis, kapasitas totalnya

sebesar 15,9 GB

DVD R =satu kali isi data, DVD-RW = diisi data berkali kali

MEMORY UTAMA

Memori Controller

MAR = Memory address register

MBR = Memory buffer register

WE = wtire enable

OE = Output Enable

Pengaksesan memori :

MAR = memuat alamat, jumlah bit menentukan jumlah maksimum, jika mar terdiri dari n->bit maka memori yang valid 0 sampai 2n-1

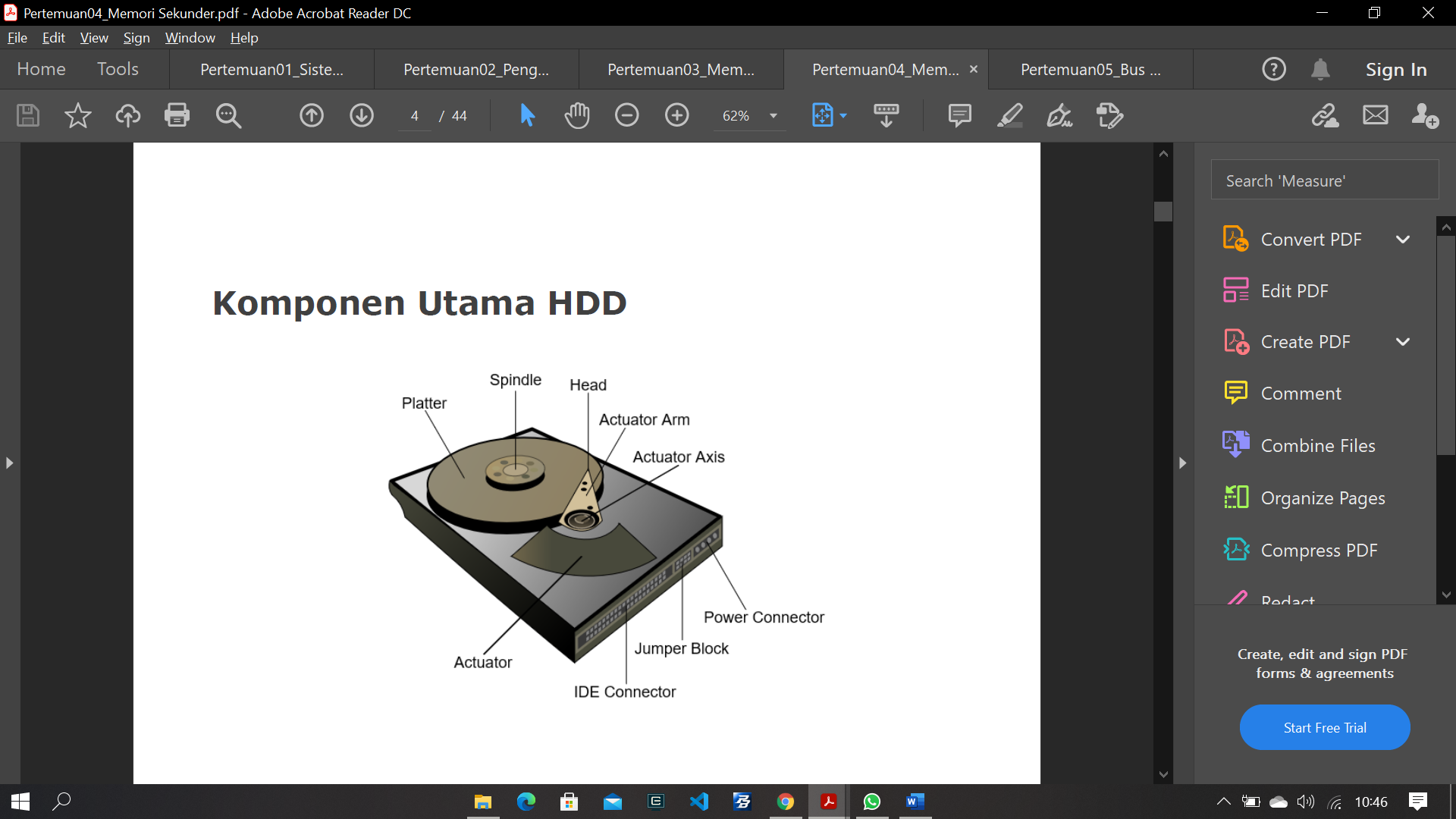
MBR = Memuat informasi yang akan dituliskan ke memori atau baru saja di baca, MBR dapat berukuran m bit, 2m bit, 4m bit, dst, dimana m = jumlah bit minimal

Memori decoder = untuk menerjemahkan alamat dikirim ke MAR sebagai x dan y

MEMORI Sekunder

Hard disk drive (HDD)/ Magentic disk

Komponen utama :

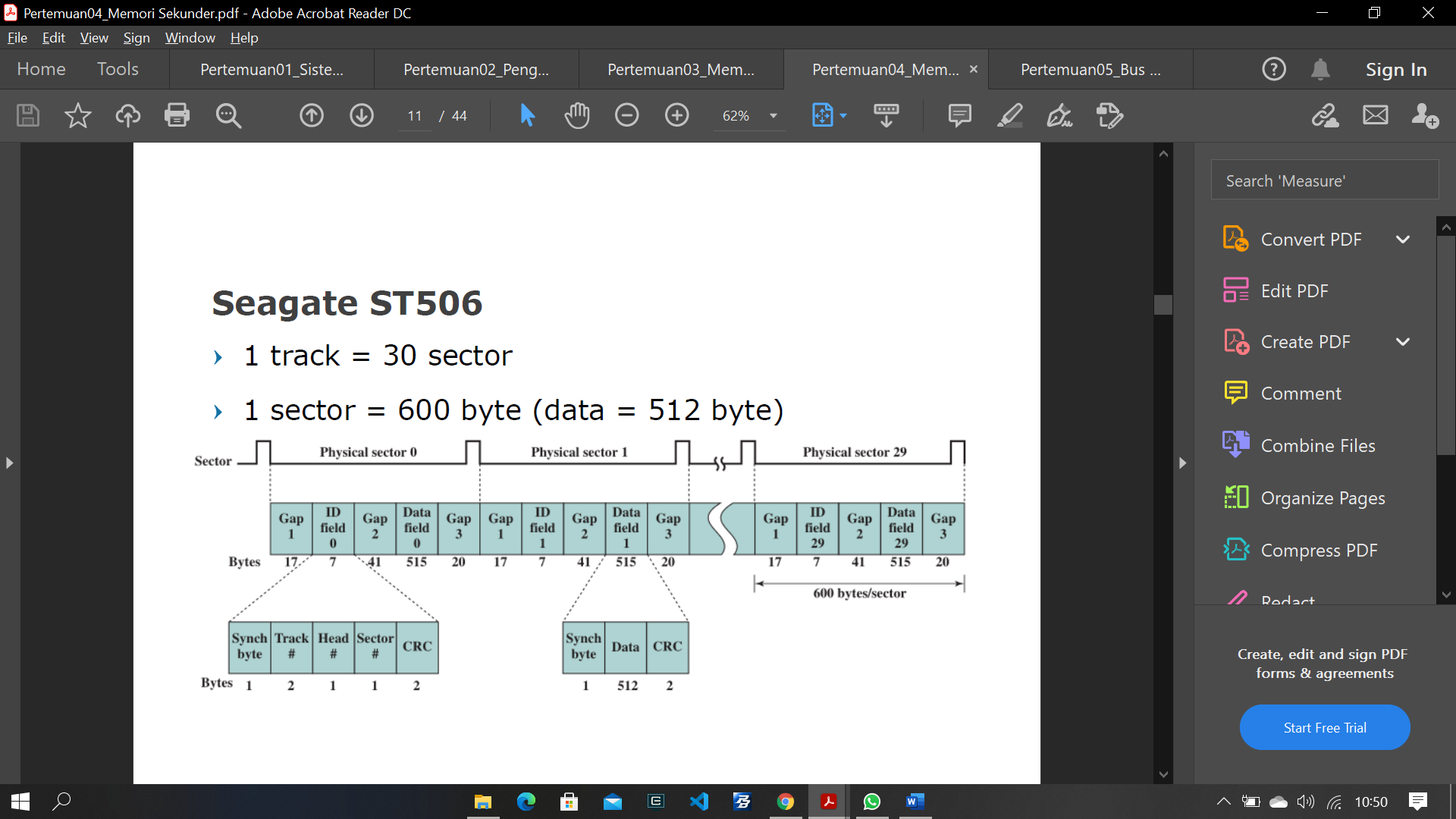


Platte : piringan alumunium/glass yang dilapisi bahan agar bisa menjadi magnet.

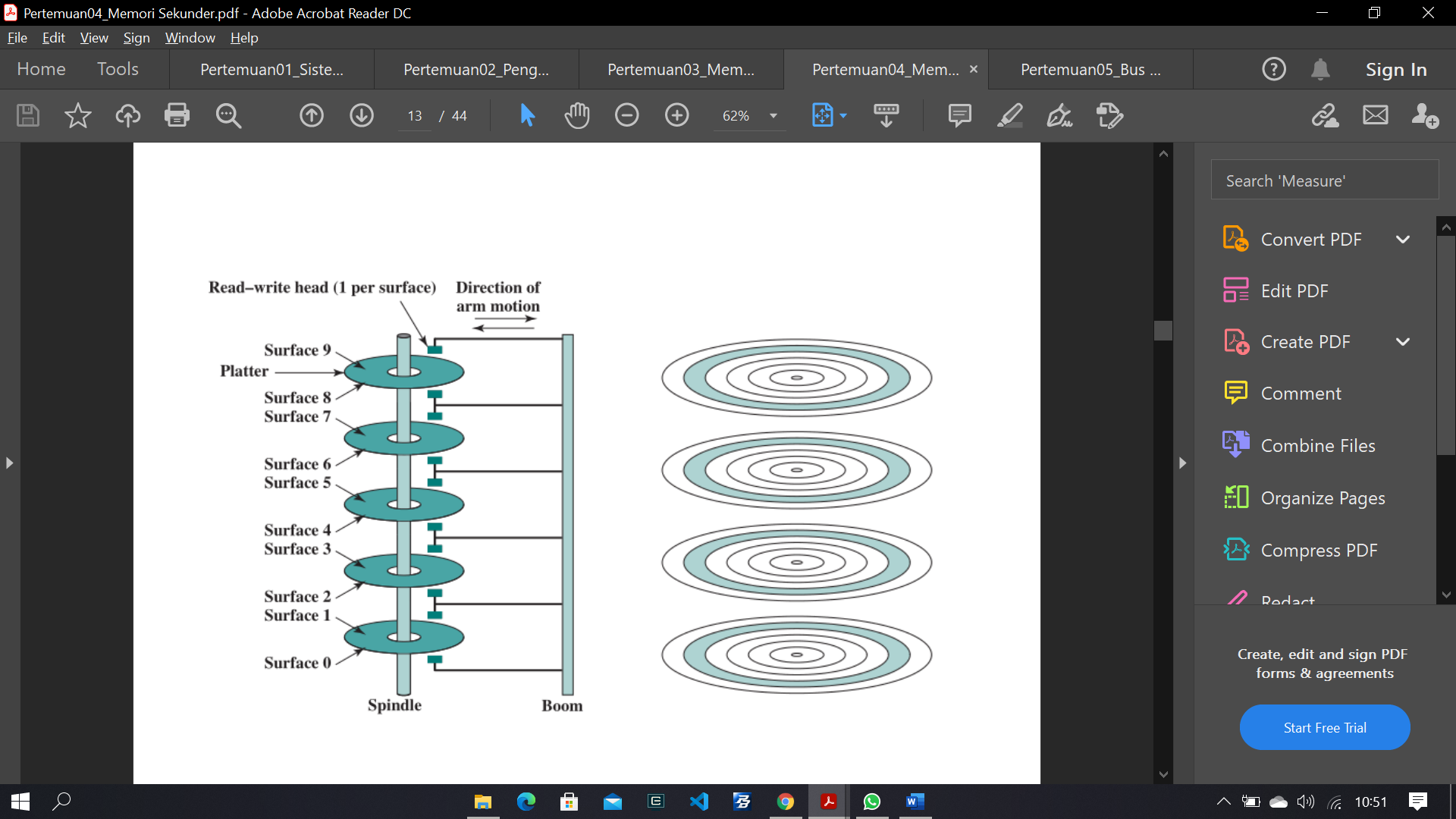
Head : untuk membaca dan menulis di platte, head adalah kotak dengan koil konduktif, head ada 2 jenis yaitu write dan read yang bekerja silih berganti (tidak dapat Bersama)

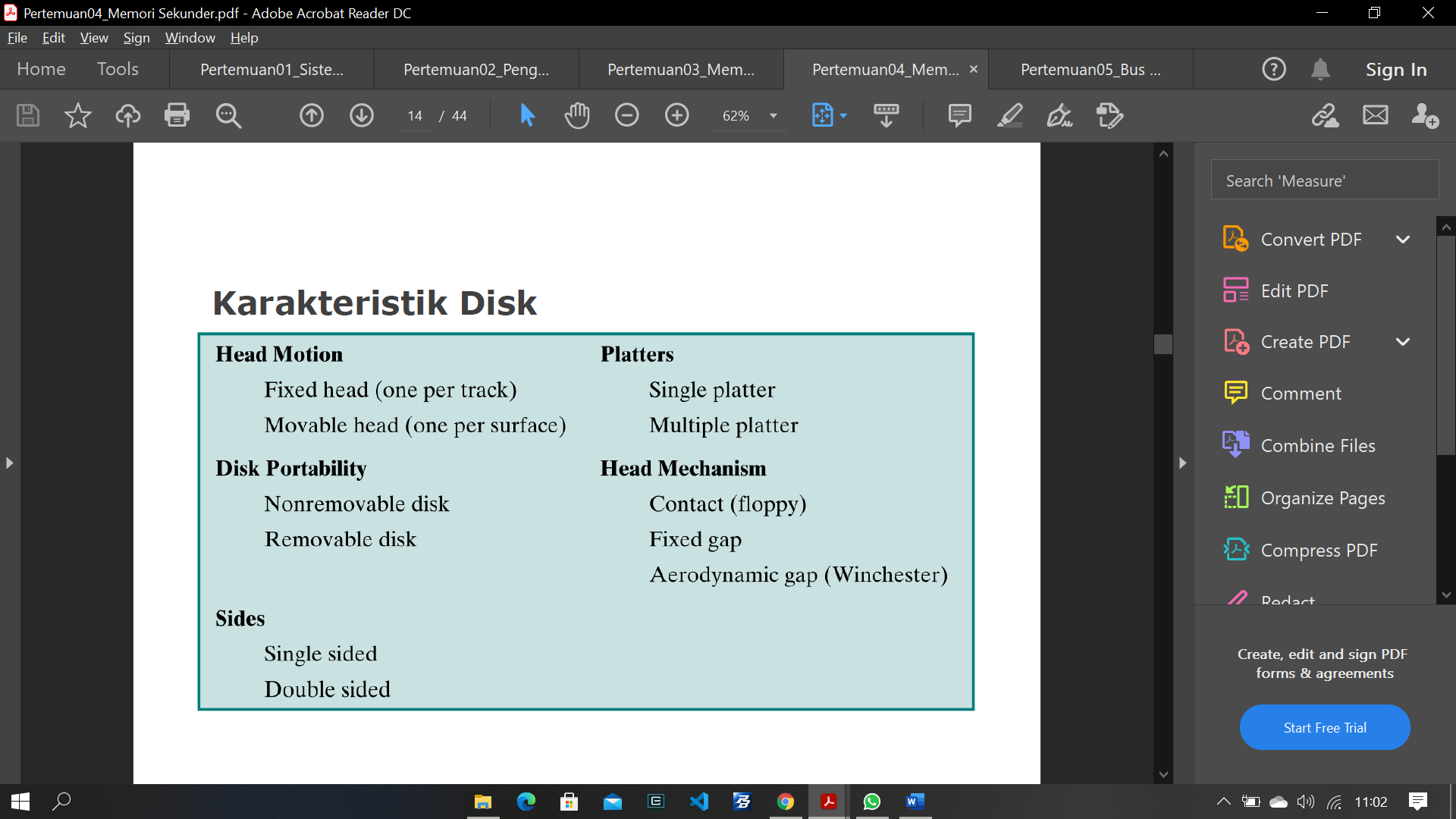
Track = track mempunyai lebar ukuran sama dengan head, track adalah jalur berbentuk lingkaran pada piringan, antar track terdapat gap yang disebut interrack gap.

Sector = data di transfer dari dan ke disk dalam satuan unit yang disebut sector, pada 1 track terdapat puluhan hingga ratusan sector, 1 sector berisi 512 byte



Cylinder : kumpulan semua track yang mempunyai posisi yang sama pada piringan





Parameter performa disk : seek time (waktu head berpindah dari satu track ke track lain, rata rata 4, 9 , 12, 15 ms), delay (waktu untuk menuju sector yang dinginkan, kecepatan rata rata -3600 s.d 20000),access time (waktu menuju posisi, rumus = avrg seek time + avrg delay) , transfer time (waktu untuk transfer data rumus = T/r =b/N. (b= jmlh byte, r = relational delay, N = jumlah byte pada track, T = Trasnfer time)) , I/O operation (membaca disk)

TOTAL : seek time + avrg relational time + transfer time + overhead time

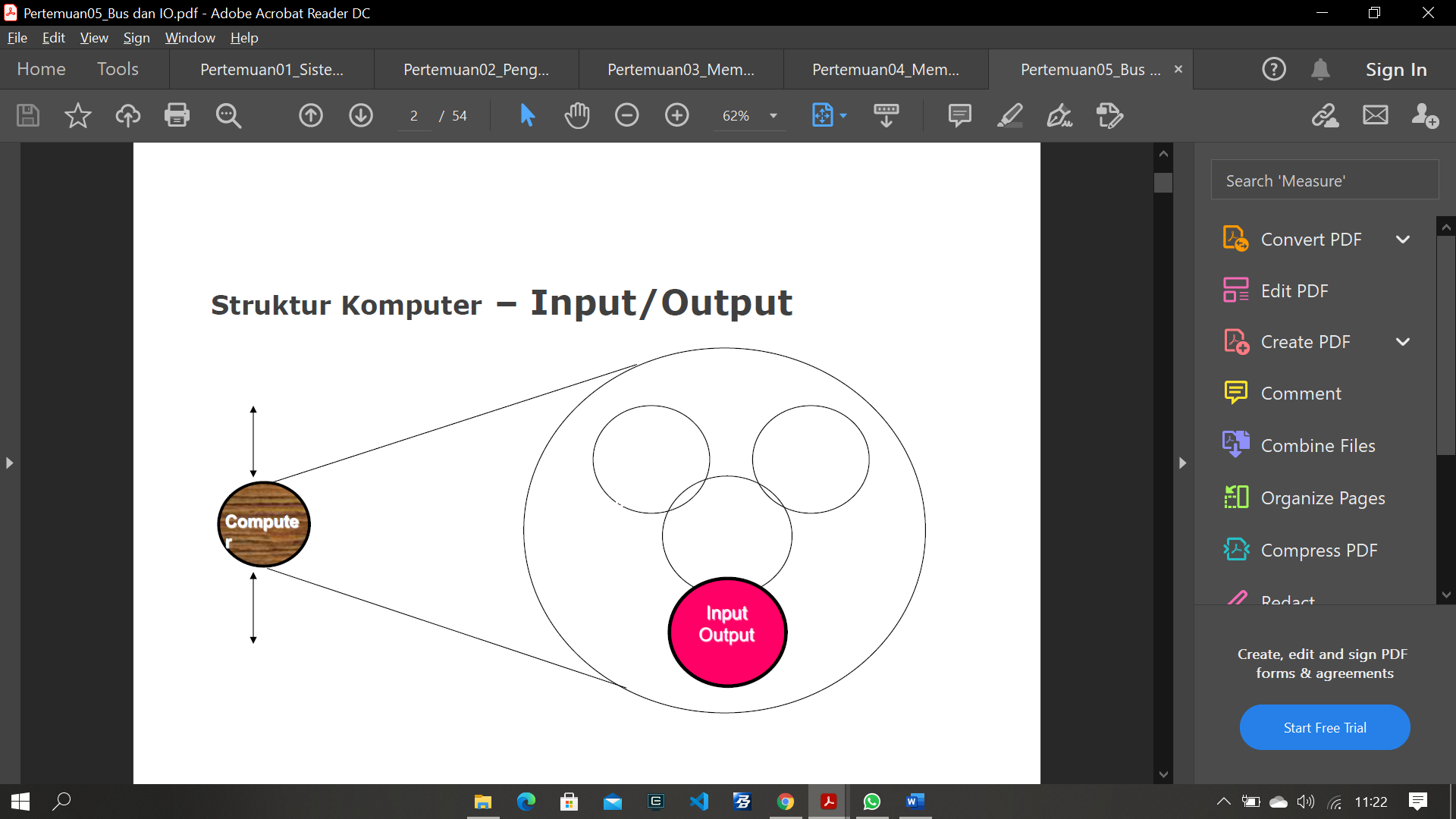
Solid state drive (SSD)

Memori yang dibuat dari komponen solid state, solid state = circuit electronic dibangun media semi conductor

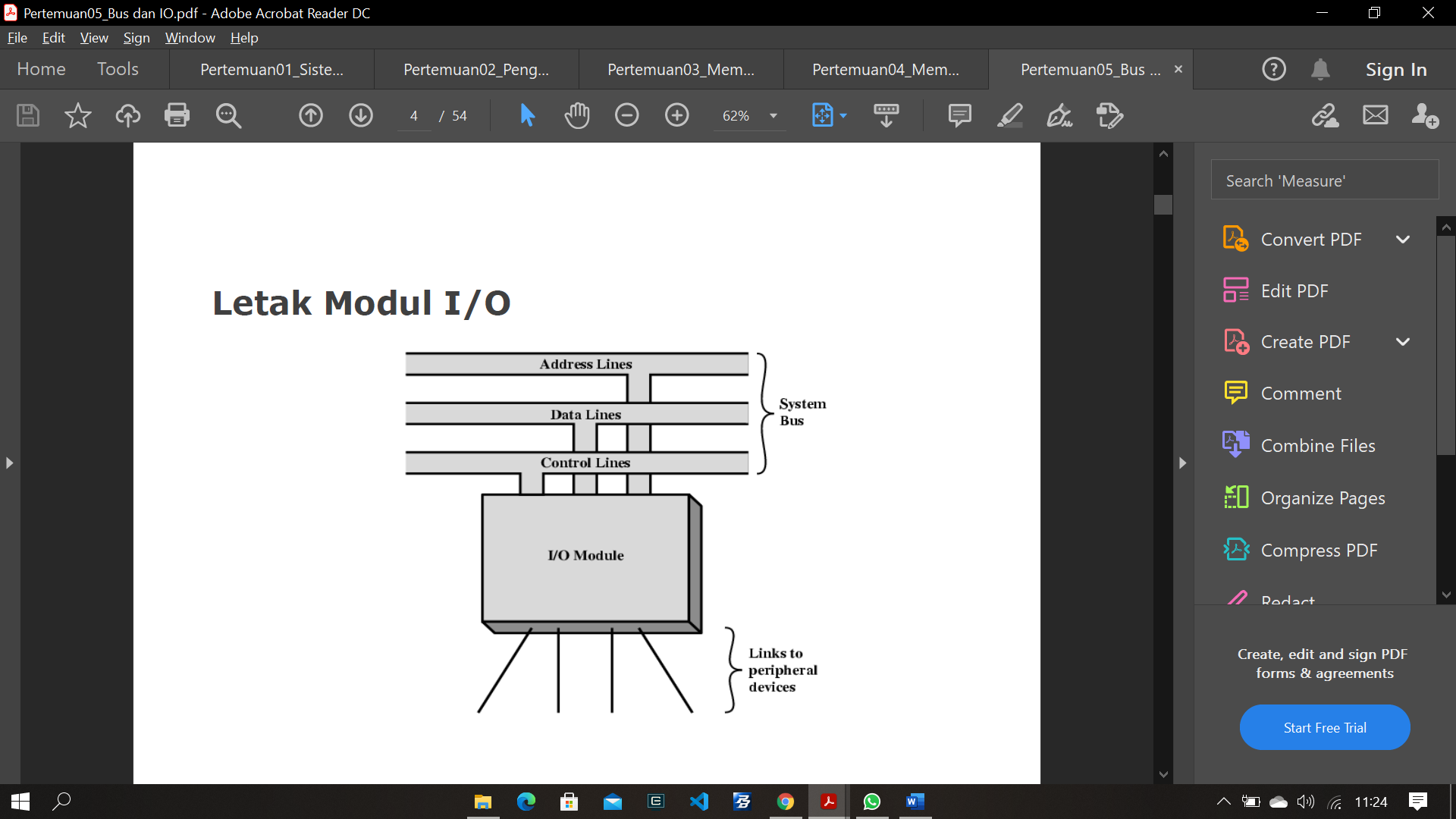
Kelebihan : I/O lebih tinggi, lebih Tangguh terhadap guncangan, life time lebih lama, konsumsi energi lebih rendah, tidak berisik.

Kekurangan : performa menurun semakin lama, mahal, setelah 100.000 write tidak bisa digunakan.

SISTEM BUS DAN I/O



Fungsi utama I/O adalah menjembatani antara CPU dan memori, untuk menjembatani satu CPU dan memori dengan beberapa peripheral. Peripheral = device eksternal yang terhubung ke I/O

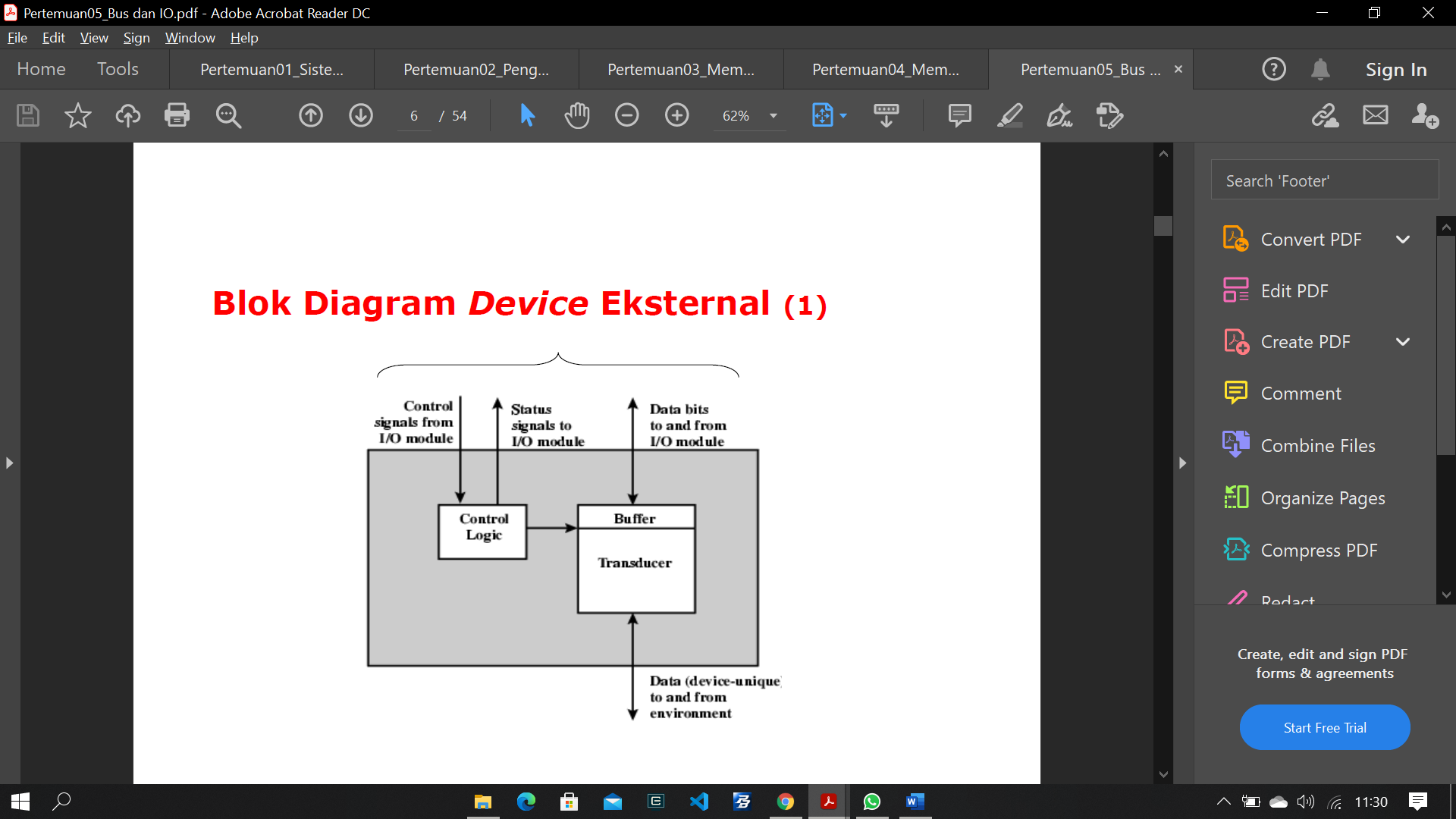


Device eksternal (Peripheral) =

Human readable = sarana komputer dengan manusia (komputer, monitor, printer, mouse, dll)

Machine readable = sarana komputer dengan device lain (harddisk, sensor, kalkulator, dll)

Communication = sarana komputer dengan komputer lain (modem)



Signal control = menentukan apa yang harus dilakukan device (input/read untuk baca, output/write untuk mengirim data ke device lain)

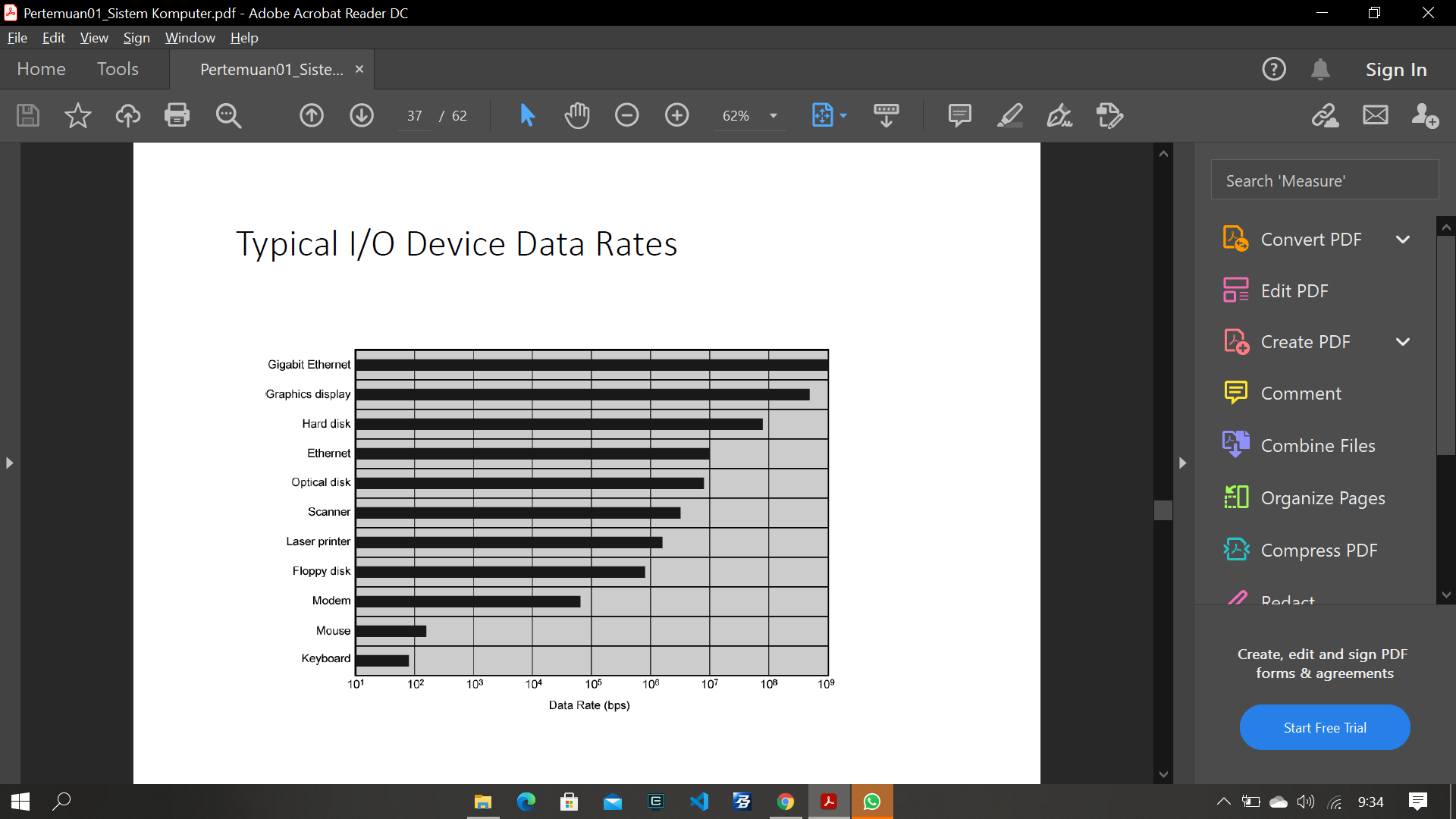
Signal status = mengirim status ke device

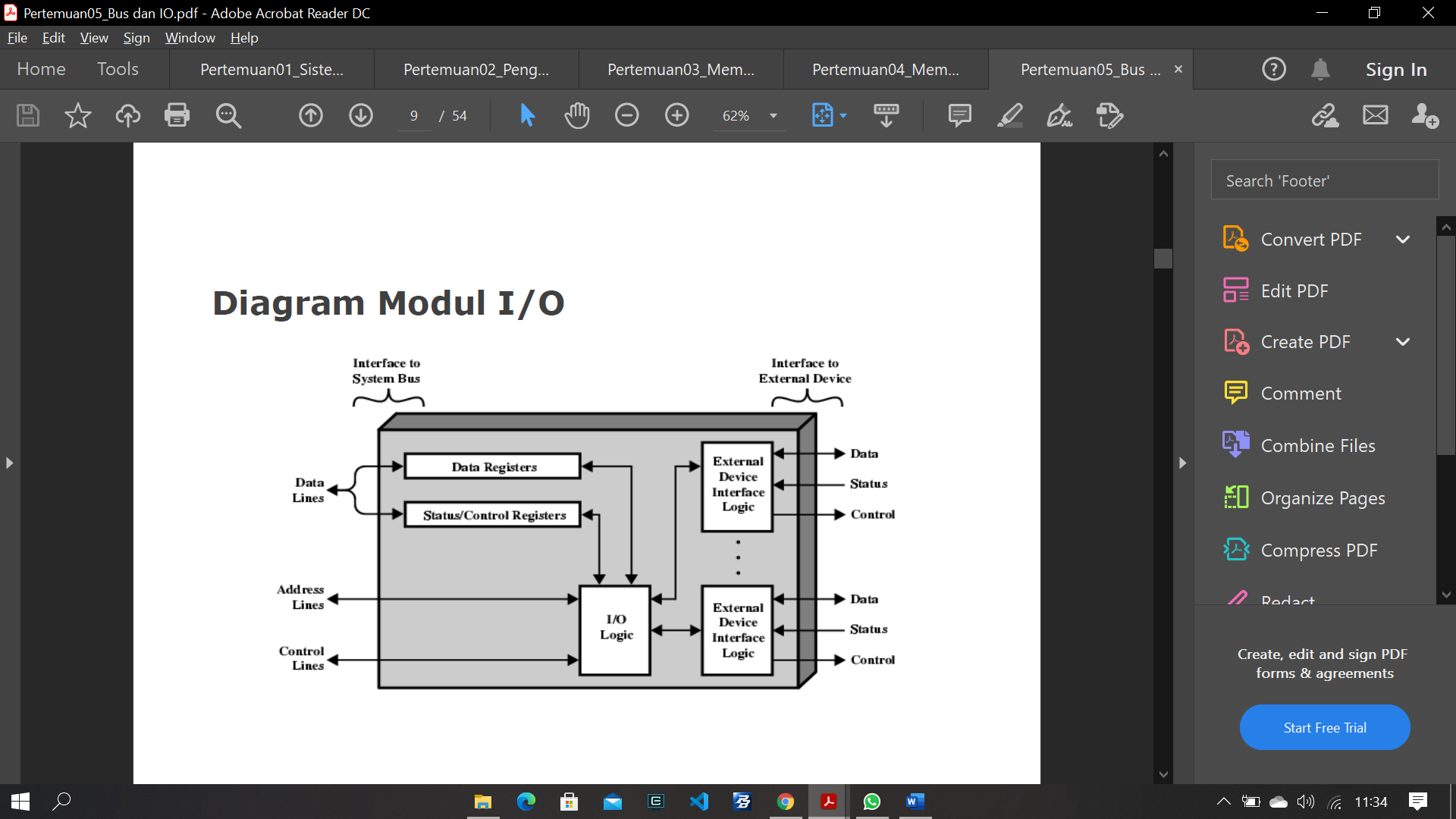
Jalur data = untuk mengirim/menerima deretan bit dari modul I/O

Control logic = menentuka aktifitas status device eksternal

Buffer = menampung data dari/ke modul I/O sementara biasa 8 s/d 16 bit

Transducer = mengubah signal elektrik, mekanik, dll ke digital dan sebalikna





Fungsi modul I/O

Control timing = mengatur aliran data antara resource internal dengan device eksternal

CPU Communication :

Sebagai media komunikasi dari CPU ke device eksternal

Device communication :

Berfungsi sebagai media komunikasi device eksternal ke cpu

Data buffering :

Modul I/O berfungsi untuk menampung data sementara baik dari cpu/memori atau dari peripheral

Error detection :

Sebagai pendeteksi kesalahan yang ditimbulkan device

**INTERRUPT:**

* a/ mekanisme utk menghentikan sementara waktu urutan eksekusi program yg normal (sequence) jika:
* \* kondisi tertentu telah terjadi
* \* ada program lain yg lbh mendesak
* utk dieksekusi
* Penyebab interrupt ada 4:
* 1. Program :
* Overflow, dicision by zero, akses illegal memori
* 2. Timer:
* \* dihasilkan oleh timer prosseor internal
* \* digunakan pada pre-emptive multi tasking
* 3. I/O:
* Dihasilkan oleh I/O controller (eksekusi telah selesai atau ada kesalahan)
* 4. Hardware failure
* Misal: memory parity error, power failure, dll
* Siklus interrupt:
* Start -> fetch next instruction-> eksekusi insturksi (jika ada intrupsi,jika tidak ada kembali ke fetch next instruction) -> cek instruksi dan jalankan instruksi (interrupt cycle - execute cycle - fetch cycle -> fetch Next Instruction)
* papale papale
* Siklus interrupt - State Diagram :
* intstruction address calculation-> Instruction fetch -> Instruction Operating decoding -> operand address calculation <-> operand fetch -> data operation -> operand address calculation <-> operand store -> interupt check (->) -> interrupt -> intstruction address calculation

1. Pengecekan interrupt ditambahkan pada siklus instruksi
2. prosesor memeriksa apakah terjadi interrupt
3. jika tidak ada, kerjakan instruksi berikutnya, jika ada (ditandai dgn signal interrupt) =>
   1. tunda program yg sdg di exe,
   2. simpan context(alamat instruksi,data,dll),
   3. PC = alamat awal routine interrupt handler,
   4. kerjakan interrupt sampai selesai,
   5. kembali context (PC = alamat saat terjadi interrupt)
   6. Lanjutkan program tertunda

Multiple Interrupts  
Ada 2 metode yang dapat digunakan:  
    Disable interrupts  
    Bila prosesor   
      
Tekni Input Oputput  
(1) Programmed I/O

* I/O terjadi pada saat program yang di dalamnya terdapat perintah I/O dieksekusi
* Eksekusi I/O terus menerus melibatkan prosesor

(2) Interrupt driven I/O

* I/O terjadi pada saat perintah I/O dieksekusi
* Sesudah perintah I/O dieksekusi ->  CPU melanjutkan eksekusi perintah lainnya -> tidak terlibat terus menerus
* CPU berperan lagi jika sudah ada interrupt dari device (modul I/O)

(3) Direct Memory Access (DMA)

* Transfer data ditangani oleh sebuah prosesor I/O khusus

baca instruksi / komen untuk i/o module (cpu -> i/o)  
      |  
      v  
baca status i/o module (i/o -> cpu)  
      |  
      v  
cek status (error condition)  
      |  
      v   
baca kata dari i/o (i/o -> cpu)  
      |  
      v  
tulis di memori (cpu -> memory)  
      |  
      v  
selesai(yes/no, kalau no keawal lagi)  
      |  
      v  
next instruction  
  
Programmed I/O (3)  
Apa yang dilakukan CPU?

* Mengirimkan alamat modul I?O (dan alamat device jika dalam modul tersebut terpasang lebih dari satu device)

Mengirimkan perintah (command):  
    Control:  
        Untuk mengaktifkan peripheral  
        Untuk menyuruh peripheralelaan sesuatu

* misal: disk berputar, head bergerak, dsb.
* Test:
  + Untuk memeriksa status device
  + Apakah ada tegangan ?, Apakah terjadi gangguan ?, dsb
* Read:
  + Untuk meminta data dari peripheral melalui modul I/O (misal baca data
  + dari harddisk)
* Write:
  + Untuk memberikan data ke peripheral (misal menulis data ke harddisk)

Programmed I/O (4)  
Jenis pengalamatan I/O:

* (a) Memory mapped I/O
  + - Ruang alamat digunakan oleh memori dan I/O bersama-sama
      * + ->Jumlah alamat untuk memori berkurang
    - CPU memperlakukan I/O seperti memori biasa
    - Tidak ada perintah khusus bagi I/O (perintah yang digunakan mirip dengan perintah untuk memori)
* (b) Isolated I/O
  + - Memori dan I/O menggunakan ruang alamat yang sama secara bergantian
      * + ->Jumlah alamat untuk I/O sama banyak dengan alamat untuk memori
    - Diperlukan select line untuk membedakan antara memori dengan I/O
    - Diperlukan perintah khusus untuk I/O
* Apa yg dilakukan CPU : mengirim perintah, kerjakan program lain, periksa keberadaan interrupt setiap akhir, kemudian tangani interupt.
* Prosesnya :
* Device controller mengirim sinyal interupt -> proessor menyelesaikan perintah eksekusi -> sinyal processor menganalisis interupt -> Processor mengirim (?) PSW dan PC ke dalam control stack -> processor mengumpulkan PC value sesuai dengan interupt -> menyimpan proses informasi -> memproses interrupt -> merestore process information -> mengganti PSW lama
* Cara menentukan asal interupt :
* (1). Bisa dalam jalurnya sendiri. Digunakan banyak jalur interrupt à Satu modul satu jalur
  + (-) Tidak praktis karena harus menyediakan sejumlah jalur khusus untuk interrupt
  + (-) Jumlah device yang dapat dipasang terbatas.
  + Kurang rekomended
* (2).  Software poll (polling) bedanya dengan interupt, sistemnya sama sebenarnya, kalau polling kita cek satu-satu kalau interupt tidak.
* (-) Lambat, karena harus memeriksa terus menerus modul I/O satu persatu Mekanisme: Jika CPU mendeteksi adanya interrupt CPU menanyakan ke setiap modul I/O untuk mengetahui asal interrupt (polling) Misal dengan mengirimkan signal TESTI/O Modul pengirim interrupt akan menjawab signal tersebut
* (3) Daisy chain atau hardware poll Digunakan satu jalur interrupt yang menghubungkan setiap modul I/O (cuman module lain dengan moduleyang lain terhubung secara serial). Modul satu dengan modul lainnya terhubung secara serial Mekanisme: Modul I/O mengirimkan interrupt melalui jalur interrupt bersama CPU mendeteksi adanya interrupt à mengirimkan signal ke sebuah modul I/O Signal akan diestafetkan dari satu modul ke modul lainnya Hanya modul yang mengirimkan interrupt yang akan memberi jawaban Jawaban berupa word (vector) yang ditaruh di jalur data. Isi word = alamat modul I/O atau identitas unik lainnya Vector digunakan sebagai pointer untuk menunjuk langsung ke device sumber interrupt sebelum menjalankan interrupt handlerà disebut vectored interrupt. Sistemnya kaya linkedlist(?)  jadi kaya nanya secara estafet sampe nemu ujungnya
* (4) Bus master Pengiriman interrupt dilakukan bila modul I/O telah ‘menguasai’ (master) bus Mekanisme: Modul I/O mengirimkan permintaan untuk menggunakan bus Arbiter memberi kesempatan kepada I/O modul à hanya satu modul dalam satu saat Modul I/O mengirimkan interrupt CPU mendeteksi adanya interrupt dan memberi respons melalui jalur acknowledge Modul I/O mengirimkan word (vector) ke jalur data.
* dalam contoh 82C59A,
* bila jumlah device lebih banyak maka 82C59A disusun secara bertingkat.

Cara kerja: 8259A menerima interrupt dari device 8259A menentukan prioritas (bila lebih dari satu device yang meng-interrupt) 8259A mengirimkan signal ke CPU 8086 (melalui jalur INTR) CPU mengirimkan respons (acknowledge) 8259A menaruh vector ke bus data CPU memproses interrupt.

* DMA kenapa di perlukan? soalnya tidak semuanya melibatkan CPU, langsung ke memori, seolah-olah tidak melibatkan CPU. Secara konsep tidak menggunakan CPU, data yg dibaca tetap di RAM tetapi yang mengontrolnya I/O.
* Karena programmed I/O dan interrupt driven I/O: Masih memerlukan keterlibatan CPU à CPU menjadi sibuk Transfer rate data terbatas
* Digunakan modul khusus (hardware) yang terhubung ke sistem bus Fungsi modul DMA: meringankan kerja processor, Dapat menirukan sebagian fungsi prosesor Dapat mengambil alih fungsi prosesor yang berhubungan dengan transfer data. DMA bekerja ? Saat prosesor sedang tidak menggunakan bus, Saat prosesor dipaksa berhenti sesaat (suspend) à siklusnya “dicuri” oleh DMA à disebut cycle stealing
* (Modul DMA hampir sama seperti processor)

Cara Kerja DMA :

* CPU mengirim data ke DMA controller (Perintah Read/Write Alamat device yang akan diakses Alamat awal blok memori yang akan dibaca/ditulisi Jumlah blok data yang akan ditransfer)
* diberi nih memori segini misalnya, kemudianvCPU mengeksekusi program lain DMA controller mengirimkan seluruh blok data (per satu word) langsung ke memori (tanpa melibatkan CPU) DMA controller mengirim interrupt ke CPU jika telah selesai

Cycle Stealing Pada DMA Transfer  
ketika CPU sedang suspend, DMA bekerja. pengambil alihan inituh bukan interrupt yah teman-teman..

* DMA controller mengambil alih bus sebanyak satu siklus DMA men-transfer satu word data Pengambil alihan bus oleh DMA bukan interrupt -> CPU tidak perlu menyimpan context CPU hanya tertunda (suspend) sesaat (satu siklus) sebelum mengakses bus Yaitu sebelum operand atau data diambil atau data ditulis Apa pengaruhnya terhadap CPU ? Memperlambat CPU, tetapi masih lebih baik daripada CPU terlibat langsung pada transfer data

konfigurasi DMA

* Konfigurasi I:  (arsitektur lama)
* Hanya menggunakan single bus DMA dan modul I/O terpisah Setiap transfer harus mengakses bus 2 kali: modul I/O ke DMA kemudian DMA ke memori -> CPU tertunda 2 kali -> lebih lambat
* procesor->DMA->I/O...->Memory

Kalau menyebabkan crash, berarti programmernya yg failed   
  
//baru sampe sini sih

* Konfigurasi II: Hanya menggunakan single bus DMA controller dan modul I/O terintegrasi Satu DMA controller dapat menangani >1 modul I/O Setiap transfer hanya perlu mengakses bus satu kali saja DMA ke memori -> CPU hanya tertunda satu kali-> lebih baik
* processor -> (DMA, I/O) -> DMA ( I/O) /\ (I/O) -> memory