NTNU
Kunnskap for en bedre verden

KANDIDAT

10611

PRØVE

TDT4160 1 Datamaskiner og digitalteknikk

Emnekode	TDT4160
Vurderingsform	Hjemmeeksamen
Starttid	30.11.2021 08:00
Sluttid	30.11.2021 12:00
Sensurfrist	21.12.2021 22:59
PDF opprettet	04.12.2021 13:39

Diverse MP

Oppgave	Tittel	Oppgavetype
ℹ	Forside	Informasjon eller ressurser
1	Diverse oppvarming	Sammensatt
2	Minnehierarki	Dra og slipp
3	Minnehierarki 2	Dra og slipp
4	Virtuelt minne	Sammensatt
5	ARM Assembler	Sammensatt
6	Cacheytelse	Fyll inn tall

Knot-0614 ASM

Oppgave	Tittel	Oppgavetype
7	Assembler-lesing Knot-60	Fyll inn tekst
8	Assemblykoding	Sammensatt
9	Mikroarkitektur 1	Sammensatt
10	Mikroarkitektur 2	Sammensatt
11	Bussdiagram	Dra og slipp
12	Diverse ytelse	Sammensatt
13	Buss-spørsmål	Sammensatt
14	Digitalteknikk 1	Sammensatt
15	Digitalteknikk 2	Dra og slipp

1 Diverse oppvarming

Velkommen til eksamen i TDT 4160.
Husk å lese oppgavene nøye, og vær ekstra nøye med formatet på svarene dine. Husk også å skrolle nedover sidene: inspera er ikke spesielt flink til å fortelle om lengre sider.

Utrykk den binære strengen 0100111100111100 i heksadesimalsk notasjon, f.eks. 0x1234.

0x4F3C

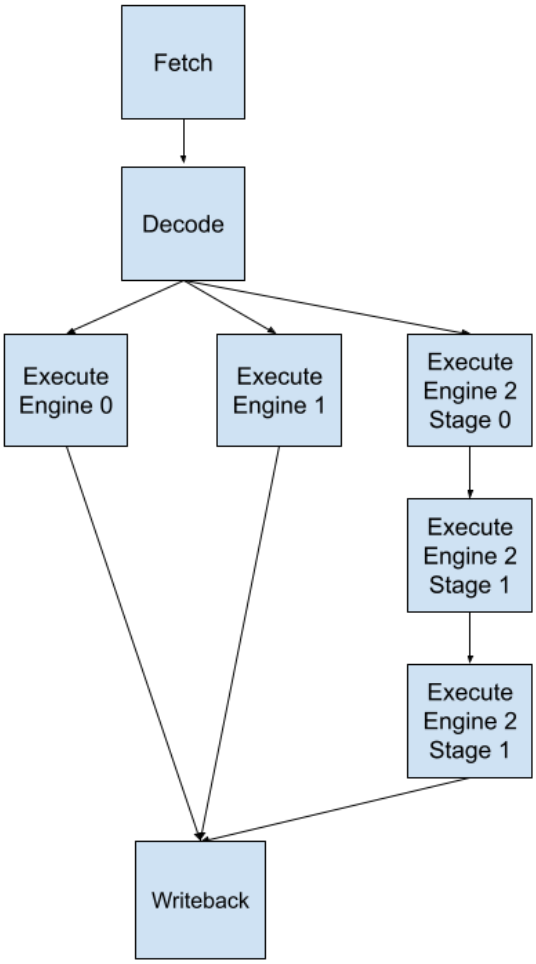
Hva er de vanlige tilstandene en datamaskin befinner seg i når den kjører instruksjoner?
Velg ett alternativ

- Decode - Fetch - Restate
- Fetch - Analyze - Compare
- Fetch - Compare - Decode
- Fetch - Decode - Execute

Hva sier lokalitetsprinsippet?

Velg ett eller flere alternativer

- [x] Det vil ofte være effektivt å hente instruksjonen og naboinstruksjoner.
- [x] Minneadresser som ble nylig brukt, vil sannsynligvis bli brukt igjen ganske snart.
- [ ] RAM er ofte lokalisert nærme CPU.



Figuren over er en arkitektskisse av en CPU. Hvilke prinsipper er brukt?

Velg ett eller flere alternativer

- [x] Samlebånd ("Pipelining")
- [ ] Flerkjerner
- [ ] SIMD
- [x] Superskalaritet

Assemblersnutten

mov r0, r3
add r3, r0, r2
inneholder følgende avhengigheter ("hazards"):
Velg ett eller flere alternativer

- [ ] RAR
- [x] WAR
- [x] RAW
- [ ] WAW

I en samlebåndsarkitektur ("pipeline") *må* det være støtte for å tømme samlebåndet ("pipeline flush") for at et program skal utføres korrekt.
Velg et alternativ

- ○ Usant
- ◉ Sant

Hva betyr det at en hurtigbuffer ("cache") er "Write Through"?

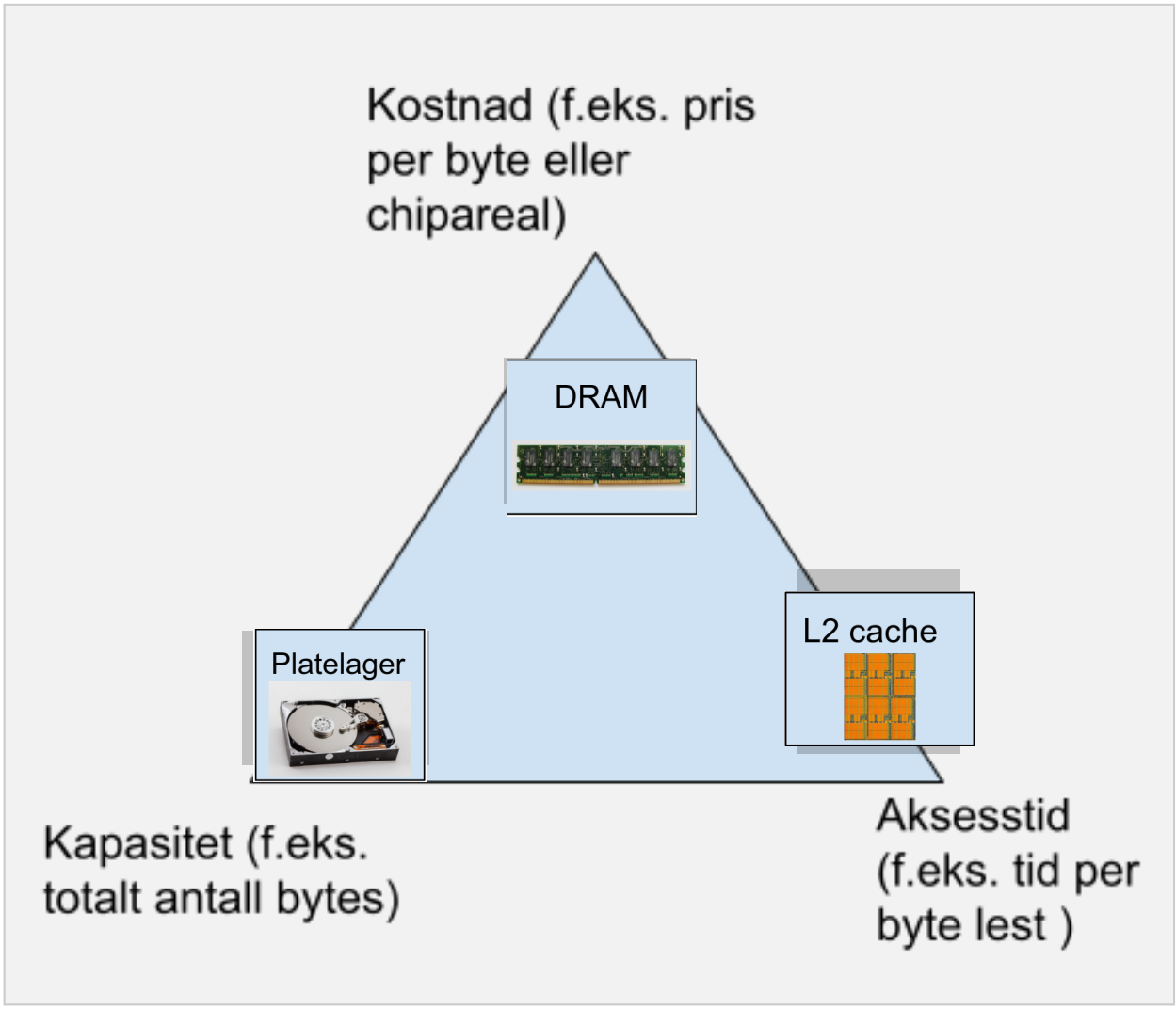
Velg ett alternativ

- ○ En cachelinje blir kun skrevet til underliggende nivå når den skal byttes ut.
- ○ En STORE-instruksjon vil umiddelbart bli utført ned hele kjeden i minnehierarkiet.
- ◉ En STORE-instruksjon vil bli umiddelbart skrevet til det underliggende nivået i minnehierarkiet.
- ○ Dersom cachen er delt mellom kjerne 1 og kjerne 2 vil kjerne 2 bli informert om skrivingeoperasjoner fra kjerne 1.

2 Minnehierarki

Trekanten viser prioriteringer. Dra de ulike minnetypene inn i riktig boks i trekanten, basert på hvilke faktorer som er **prioritert** for de ulike mediumene.

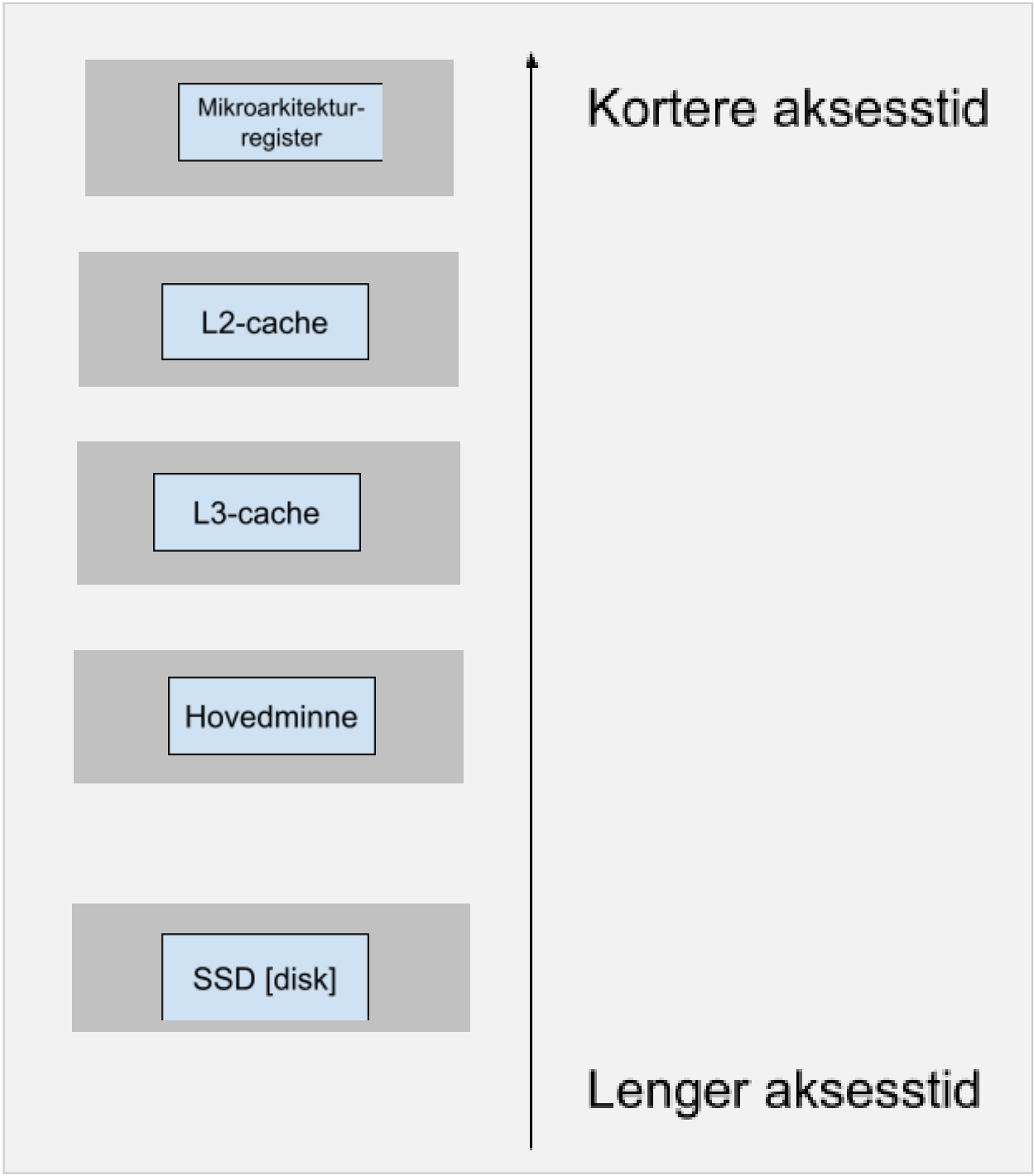
Velg ett alternativ:



3 Minnehierarki 2

Dra og slipp for å sortere minnetypene etter aksesstid. Kortest aksesstid på toppen og lengst aksesstid på bunnen.

Velg ett alternativ:



4 Virtuelt minne

De følgende oppgavene handler om virtuelt minne.

Anta 16 bits virtuelle adresser. Anta 2kB store sider ("pages"). Anta at et 8kB fysiske minne er delt inn i 2kB-store rammer ("frames").

Studer de følgende 16-bits *virtuelle* adressene, og finn de tilsvarende fysiske adressene, gitt sidetabellen:

Virtuell side	Fysisk minne ramme ("frame")
0	0
1	3
2	0
3	2
4	0
5	1
6	0
7	2

Svarene skal gis som en 13-bits binærstreng *uten mellomrom, f.eks. 0011001100110011*

1: 00010 00101110001 :

0000101110001

2: 00111 01101000011 :

1001101000011

3: 00011 00111000001 :

1000111000001

Anta at de tre oppslagene til minne skjer i den gitte rekkefølgen. Når vil det oppstå en "page fault"?

Velg ett alternativ

- Etter den første aksessen
- Etter den andre aksessen
- Etter den tredje aksessen
- Det er ikke nok informasjon til å svare på spørsmålet ut fra gitt tabell.

Hvor ligger sidetabellen ("page table") lagret i et moderne datamaskinsystem?

Velg ett alternativ

- Cache
- RAM
- En egen enhet i CPU
- Disk

5 ARM Assembler

En stor del av faget har vært å programmere assembler i ARM-Assembler.

Vedlagt ligger et "cheat sheet" for instruksjonssettet til ARM.

Din oppgave blir å fylle inn verdiene til de forespurte registerene etter beste emne.

Anta følgende verdier i minnet:
0x000000001: 0x00000042
0xABABABAB : 0x00000041
0xDEADBEEF: 0xDEADBEEF
0xFFFFFFBD: 0xABABABAB

Registerverdiene er alle 0, bortsett fra
r0 = 0x0000FFFF
r1 = 0xFFFF0000

Skriv svarene med heksadesimalsk notasjon, og fyll ut ordene fullstendig: vi bruker 32 bits ord (8 heksadesimalske siffeR). F.eks. 0xBADF000D

Koden starter her:

---

and r2, r1, r0

r2: 0x00000000

movs r0, #1
add r3, r0, r2

r3: 0x00000001

ldr r3, [r3]

r3: 0x00000042

cmp r3, #41
ble .foo
add r0, r3, #1
.foo:

r0: 0x00000043

6 Cacheytelse

Et spesielt slemt program har en L2-cache-hit-ratio på 0.4 (dvs at det treffer L2 cache 40% av minneaksesser). Programmet består av 80% LOADs fra minnet og resten aritmetikk.

Hvor stor del av minneaksessene går til neste nivå i minnehierarkiet? *Svar i hele prosent. Rund ned.* 0.48 %.

Anta at en L2-cache-aksess tar 10 ns.
Nivået under L2-cachen bruker 200ns på å slå opp ordet på den forespurte adressa.

Anta at maskinen kjører 100 instruksjoner av programmet og at det oppfører seg som beskrevet over. Hvor lang tid bruker maskinen totalt på LOADS?

9920 ns

7 Assembler-lesing Knot-60

Knot-60 er en hjemmesnekret mikroarkitektur.

Selve mikroarkitekturen er beskrevet i figur 1 i vedlegget , sammen med instruksjonssettet (ISA). **Les denne nøye: Knot-60 har mange merkelige, og antageligvis på kanten til uansvarlige, designvalg.**

Din oppgave er å utføre følgende assemblerprogram. Der det er en tom boks skal fylle inn registerverdie(ne) som du forventer å finne. **Du skal utelukkende bruke heksadesimalsk notasjon for å fylle inn svarene dine, og du skal bruke hele bytes som fylles ut til hele bit-bredden. F.eks. ikke 0x0, men 0x00.**

Anta at alle register starter med verdien 0.
Det er koblet et minne til CPU-en.
Minnet er satt til 0, med unntak av følgende adresser:

0x0002: 0xFF
0x0003: 0x00 //r
0x0004: 0x50 //g
0x0005: 0x9E //b
0x0006: 0x70
0x0007: 0x7F

Programmet starter på adresse 0x0100. For hver instruksjon må du fylle inn noen utvalgte registerverdier. Tallet forran kolon er instruksjonsnummer, og viser relativ adresse (offset) inn i koden der instruksjonen befinner seg. F.eks. er instruksjon merket 1: på adresse 0x0101. **Du skal utelukkende bruke heksadesimalsk notasjon for å fylle inn svarene dine, og du skal bruke hele bytes som fylles ut til hele bit-bredden. F.eks. ikke 0x0, men 0x00.**

0: MOV #0x04
1: MOV R0, R3

Reg_0: 0x04

Reg_3: 0x04

ADDR_LO: 0x01

2: FLD Reg_1, [Reg_0]

Reg_1: 0x50

ADDR_LO: 0x04

ADDR_HI: 0x01

3: INC Reg_0

Reg_0: 0x05

4: FLD Reg_2, [Reg_0]

Reg_2: 0x9E

5: MOV #0x6
6: MOV Reg_0, Reg_3
7: FLD Reg_0, [Reg_0]

Reg_0 0x70

8: MOV #0x2
9: FLD Reg_3, [Reg_3]

Reg_3: 0xFF

//loop:
10: STR Reg_2, [Reg_0] //write R val.

ADDR_LO: 0xFF

ADDR_HI: 0x70

11: DEC Reg_3

Reg_3: 0xFE

12: FST Reg_1, [Reg_3] //write G val

ADDR_HI 0x01

13: DEC Reg_3
14: DEC Reg_3
15: DEC Reg_3

Reg_3: 0xFB

//compare
16: CMP #0x0
17: BZ #-0x7

PC_HI 0x00

PC_LO 0x0F

8 Assemblykoding

De følgende oppgavene omhandler instruksjonssettet til Knot-60.

Hvilket format har instruksjonen "**MOV #0x2**"?

Velg ett alternativ

- D
- C
- B
- A

Vi ønsker å legge til en instruksjon for å flytte fra et register til Reg_3: **"MOV Ri".** Hvilket format passer det å benytte seg av her?

Velg ett alternativ

- D
- B
- A
- C

Instruksjonen **MOV Reg_3, Reg_3** er mulig å utføre direkte på mikroarkitekturen Knot-60.
Velg et alternativ

- Usant
- Sant

Instruksjonen **ADD Reg_1, Reg_1** er mulig å utføre direkte på mikroarkitekturen Knot-60.
Velg et alternativ

- Sant
- Usant

9 Mikroarkitektur 1

Kontrollenheten til Knot-60 er en tilstandsmaskin, der man i enkelte tilfeller må vente på at en instruksjon skal bli ferdig å utføre. I tillegg, siden Knot-60 kun har 1 minnebuss, må samme buss benyttes både for å hente instruksjoner og data.

I de følgende oppgavene skal du anta at instruksjonen allerede befinner seg i instruksjonsregisteret til kontrollenheten i Knot-60.

Bruk figur 1, tabell 3 og 4 for å fylle ut tabellene under. Bruk binære verdier.

Merk at immediate-verdier fra instruksjonen kommer på B-bussen rett fra instruksjonsregisteret.

For alu_func fyller du inn 3 siffer, f.eks. 101. For hver av de andre fyller du inn kun ett siffer, f.eks. 0.

ADD R0, R1

alu_func	001
reg_0_oe	0
reg_1_oe	1
reg_3_oe	1
reg_0_we	1
reg_1_we	0
reg_3_we	0

MOV #0x3

alu_func	100
reg_0_oe	0
reg_1_oe	0
reg_3_oe	0
reg_0_we	0
reg_1_we	0
reg_3_we	

	1

BZ Reg_0

alu_func	100
reg_0_eo	1
reg_1_eo	0
reg_3_eo	0
reg_0_we	0
reg_1_we	0
reg_3_we	0
reg_addr_lo_we	1
pc_lo_we	1
reg_addr_hi_we	0
reg_addr_hi_we	0
pc_lo_inc_mux	1
pc_hi_inc_mux	0

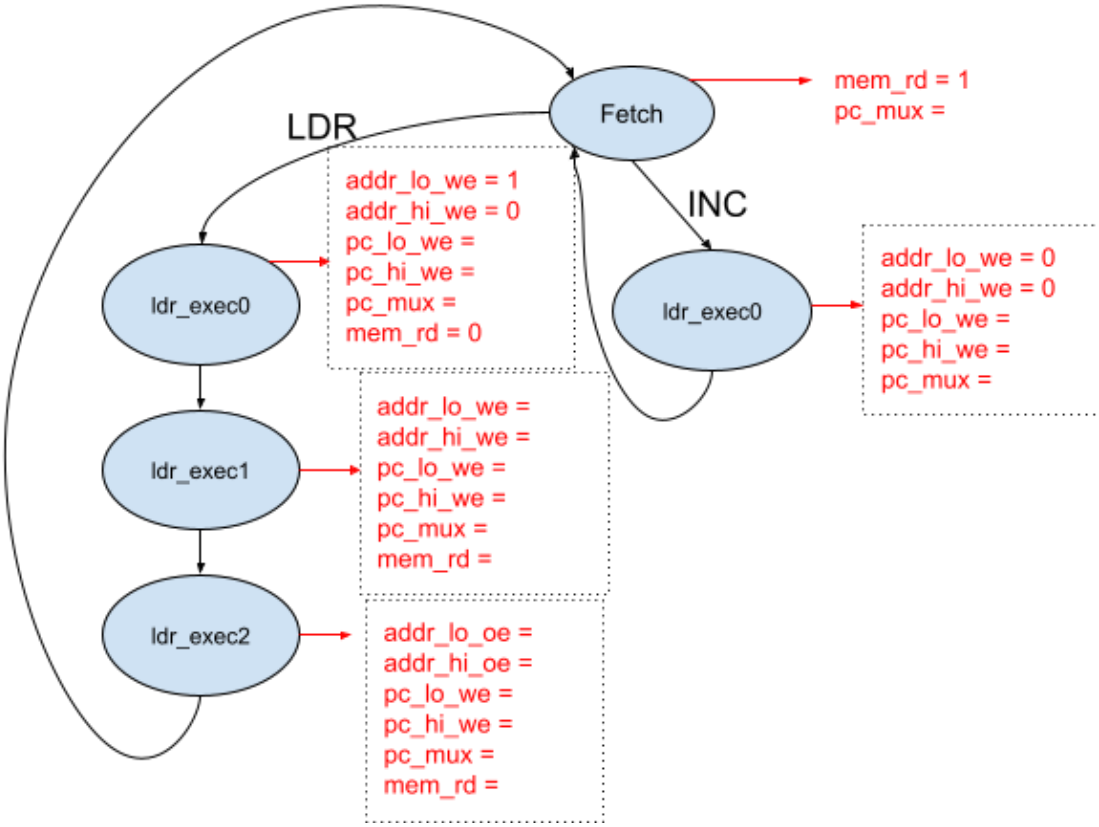
INC Reg_3

alu_func	110
reg_0_oe	0
reg_1_oe	0
reg_3_oe	1
reg_0_we	0
reg_1_we	0
reg_3_we	

	1

10 Mikroarkitektur 2

Kontrollenheten til Knot-60 er egentlig implementert som en tilstandsmaskin. Under ser du et utdrag av denne, og signalene som kontrollerer minnesystemet. Merk at det er en rekke signaler som du brukte i forrige oppgave som ikke er med i tilstandsmaskinen sine outputs. Se bort fra disse i denne oppgaven.



Hvor mange klokkesykler bruker Knot-60 på å hente og utføre en **LDR**-instruksjon? 4

Diagrammet over viser utgangssignal til tilstandsmaskinen, som blir brukt som kontrollsignal i Knot-60.
Verdiene til flere kontrollsignaler mangler. Fyll inn 0 eller 1 for hvert kontrollsignal og tilstand. Dersom signalet er irrelevant for tilstanden, sett kontrollsignalet til 0.

Fetch:

pc_lo_we: 0 pc_hi_we: 0 pc_mux = 0

LDR_exec0:

pc_lo_we: 0 pc_hi_we: 0 addr_lo_we 1 addr_hi_we 0 pc_mux 0

mem_rd 1

LDR_exec1:

pc_lo_we: 0 pc_hi_we: 0 addr_lo_we 0 addr_hi_we 0 pc_mux 0

mem_rd 1

LDR_exec2:

pc_lo_we: 0 pc_hi_we: 0 addr_lo**_oe** 1 addr_hi_**oe** 1 pc_mux 1

mem_rd 1

INC_exec0:

pc_lo_we: 1 pc_hi_we: 0

Studer mikroarkitekturdiagrammet, ISA og tilstandsmaskinen. Hvor mange klokkesykler bruker Knot-60 *minst* på å hente og utføre en INC [Ri]-instruksjon? 4

11 Bussdiagram

Minnebussen til Knot-60 er koblet til et synkront minne. Dette betyr at når Knot-60 legger en adresse på adressebussen og hever mem_rd (Rd i diagrammet under) så vil data være klare fra minne på den neste stigende klokkeflanken. *Det tar omtrent 1/2 klokkeperiode fra Rd går høy, til data er klar på databussen fra minnet: i mellomtiden er signalene på databussen fra minnet* ***udefinert / ukjent.***

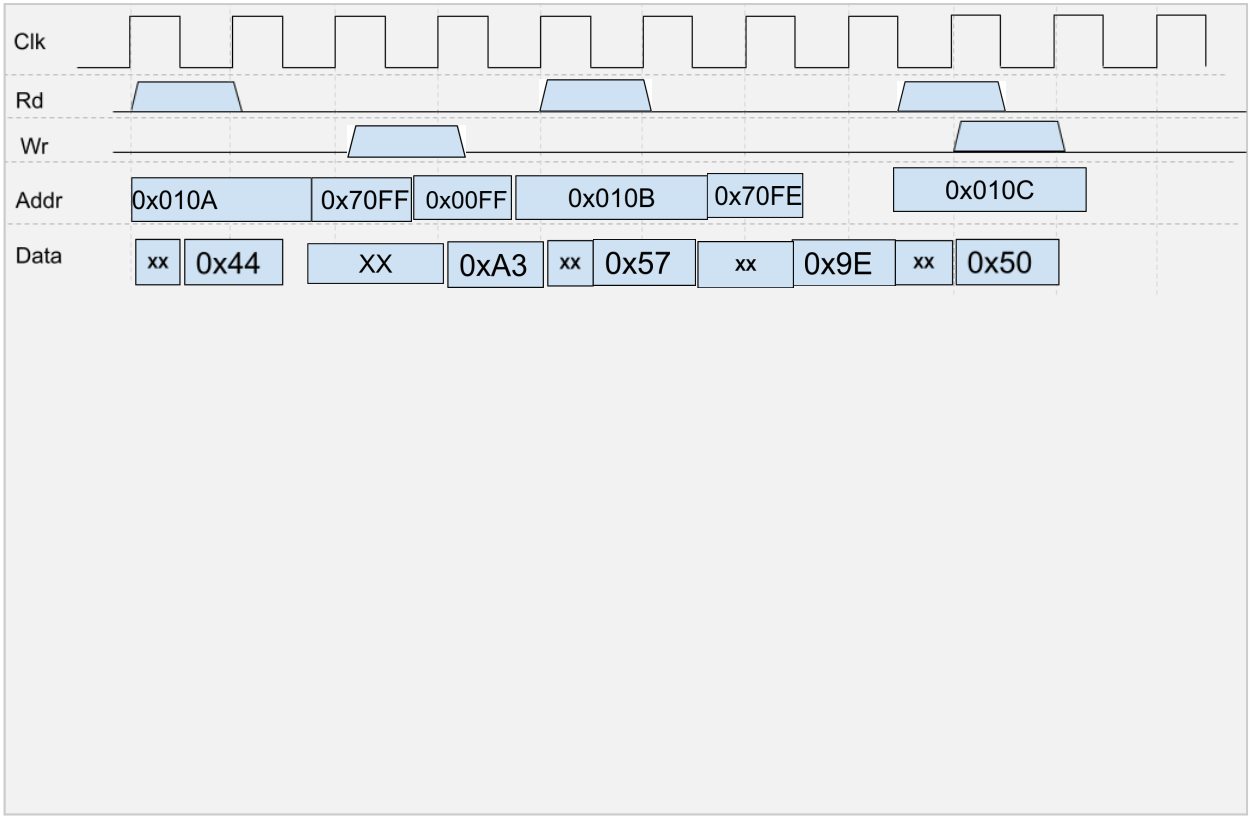
På samme vis er det slik at dersom Knot-60 legger en minneadresse på adressebussen og et dataord på databussen, og Knot-60 setter mem_wr (Wr i diagrammet under) høyt, så vil ordet bli "fanget" av minnet på den neste stigende flanken.

Programmet starter på linje 10 i programmet du "kjørte" i oppgave 7: "Assembler-lesing Knot-60". Programmet ligger vedlagt.

Rd og Wr er 1-bits signal, men for synligheten sin skyld har de blitt farget blå. Den første sykelen på bussen er altså Rd høy i 1 klokkesykel.

Strev etter nøyaktighet når du plasserer signalene, relativ til klokkeflankene.

Under ser du et utdrag fra bussen mellom hovedminnet og Knot-60. Flytt delene av bussdiagrammet til riktig sted på bussen. Hint! Start med å plassere den første minneadressen sammen med det første høye "Rd"-signalet.



12 Diverse ytelse

Bussen i forrige oppgave er en synkron buss.
Velg et alternativ

- (•) Sant
- ( ) Usant

Anta at klokken til Knot-60 har en frekvens på 100 MHz.

Hvor lang tid i *nanosekunder* tar en busstransaksjon som varer i 8 klokkesykler?

80

Hvor lang tid i nanosekunder tar alle busstransaksjonene som vist i forrige oppgave?

110

Programmet i Oppgave "Assemblerlesing" skriver RGBA-verdier (4 byte) til videominnet, byte for byte. Hver verdi blir tolket som intensiteten til henholdsvis Rødt, Grønt og Blått.

Videominnet er videre koblet til en (bitteliten) skjerm med oppløsning 8x8 piksler, der hver piksel består av 4 bytes. 25 ganger i sekundet blir deler av "Video Ram" lest ut av skjermen. Det betyr at skjermen har en oppfriskningsrate på 25 Hz. Knot-60 må fylle deler av videominnet ("Video RAM") med data før skjermen kan oppdateres (viss ikke oppleves et fenomen vi kaller "Tearing"). I dette svaret skal du anta at skjermen leser ut data fra minnet uten noe tidsforbruk.

Hvor lang tid i *sekunder* har Knot-60 på seg for å fylle videominnet? 0.04

Studer løkka som fyller videominnet i oppgaven "Assemblerlesing" (oppave 5) og bruk sykeltallene som gitt av instruksjonssettet til å svare på følgende oppgave.

Hvor mange klokkesykler bruker Knot-60 på en iterasjon av løkka, inkludert hopp (branch)?

19

Hvor lang tid i *nanosekunder* bruker Knot-60 på å fylle videominnet? 190

Hvor mange piksler (rund ned) kunne Knot-60 klart å fylle, om vi ser bort fra eventuell ekstra instruksjoner som måtte bli brukt for å kontrollere løkkene (loops)? 13473684

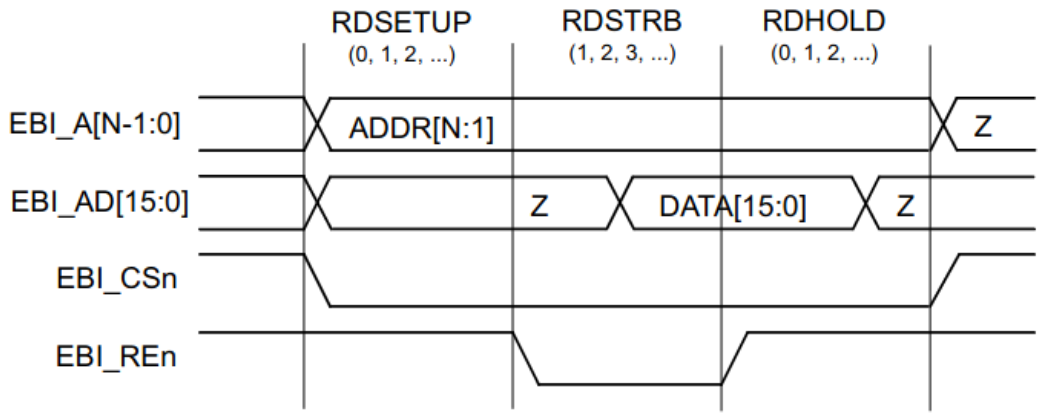
Anta et program som kontinuerlig skriver til det eksterne minnet med STR (ignorer hopp) og en CPU og bussklokke på 33MHz.

Hvor mange byte i sekundet kan du få ut av bussen? Svar i bytes/s 8250000

---

13 Buss-spørsmål

De følgende oppgavene relaterer til følgende bussdiagram.



Dette er en seriell buss:

Velg ett alternativ

- Kun for skriving.
- Korrekt
- Ikke korrekt

Hvor bredt er dataordet som overføres på denne bussen? 16 bits.

Bussen er asynkron:

Velg ett alternativ

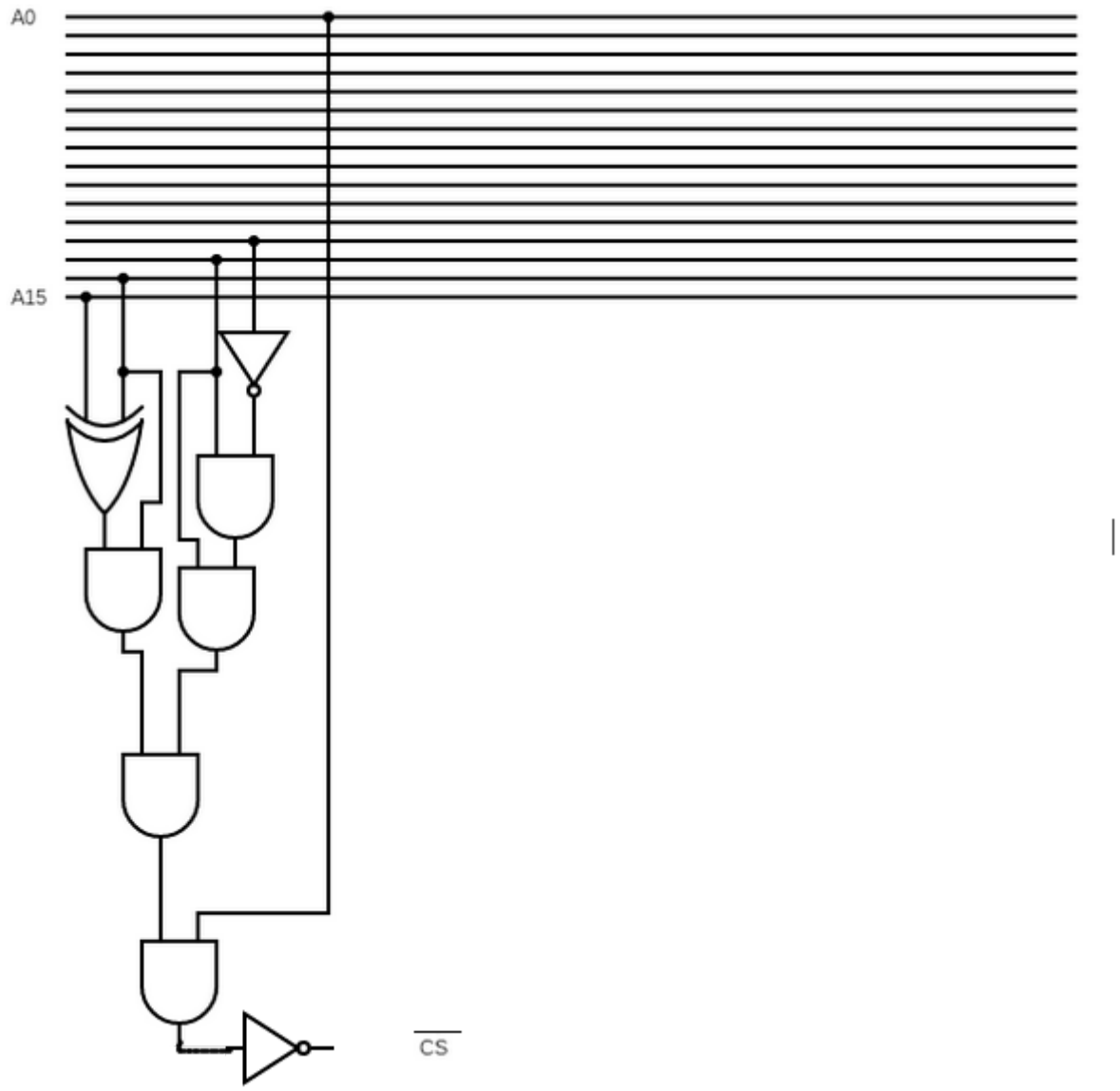
- Korrekt
- Ikke korrekt

Bussen benytter "Three way handshake"

Velg et alternativ

- Usant
- Sant

14 Digitalteknikk 1



Figuren viser et forsøk på å dekode en adresse på adressebussen.

Anta at CS er aktiv lav. Anta videre at hver av delene i minnekartet har 1 CS linje. Hvilke enhter i minnekartet kan du skru på med denne dekodingslogikken?

Velg ett eller flere alternativer

- [x] Pixel Processor Config
- [x] General RAM
- [x] Pixel Processor Status Register
- [x] Pixel Processor Sprite Memory
- [ ] Video RAM

Hva er **adresserommet** som denne logikken aktiverer?
Angi svaret på heksadesimalsk form med 4 siffer etter 0x, f.eks. 0x1234

Fra 0x6001 Til 0x6FFF

15 Digitalteknikk 2

Det viser seg at det kan være greit å ha anledning til å koble til et tastatur til datamaskinen. En avbruddslinje (interrupt) blir koblet til CPU, og ekstra logikk blir innført i kontrollenheten for å agere på avbruddet (interrupt).

Din oppgave blir nå å lage en unik dekoding av adresse 0xC901 til et *aktivt lavt* chip-select (CS) signal.

Dra og slipp de logiske portene til riktig slippsone (røde firkanter).

Velg ett alternativ: