**NORGES TEKNISK-
NATURVITENSKAPELIGE UNIVERSITET
INSTITUTT FOR DATATEKNIKK OG INFORMASJONSVITENSKAP**

Faglig kontakt under eksamen:
Institutt for datateknikk og informasjonsvitenskap, Gløshaugen
Haakon Dybdahl 735 90542 / 92259991

EKSAMEN I EMNE TDT4160 DATAMASKINER GRUNNKURS

AUGUST 2004
KL. 09.00 – 13.00

Hjelpemidler: D – Ingen trykte eller håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt.

Sensuren faller 2004.
Resultater gjøres kjent på http://studweb.ntnu.no/ og sensurtelefon 810 48 014.

Totalt Antall sider (inklusive vedlegg): 11

Vedlegg: Manual DARK load/store-arkitektur

Prosentsatser viser hvor mye hver oppgave teller innen settet.

Lykke til!

Oppgave 1 – Flervalgsoppgaver (32%)

Bruk svararket bakerst i oppgaveteksten for å svare på denne oppgaven. Du kan få nytt ark av eksamensvaktene dersom du trenger dette. Poengberegning per spørsmål: Riktig svar 2 poeng, galt svar gir -1 poeng, blankt svar gir 0 poeng. NB! Det er ikke mulig å gardere ved å krysse av flere alternativer. Dette gir i så fall 0 poeng. Kun ett alternativ er korrekt på hvert spørsmål.

1) Hva er forholdet mellom en mikrooperasjon og en instruksjon?

a) Flere mikroinstruksjoner kan bygge opp en mer kompleks instruksjon

b) Flere instruksjoner bygger opp en mikroinstruksjon

c) Kjært barn har mange navn, mikroinstruksjon og instruksjon er det samme

d) Mikroinstruksjon er mer effektivt fordi strukturen er mer tilpasset hurtiglager

2) Hvilke av følgende oppgaver er normalt IKKE avhengig av verdien til statusregistret?

a) Betinget hopp

b) Ubetinget hopp

c) Addisjon

d) Subtraksjon

3) Enkelte maskiner har noe som heter segmentregister, hva brukes det til?

a) Peker på riktig register som skal leses av i en registerfil

b) Brukes av hurtiglager for å vite hvor data skal ligge i hovedlager

c) Holder øverste bit i en adresse for å få større adresserom

d) Holder nederste bit i en adresse for å vite relasjon til hurtiglager

4) Hva er IKKE riktig om samlebånd?

a) Samlebånd øker ikke hastigheten for hver enkelt instruksjon isolert sett

b) Prosessorer med samlebånd benyttes i så godt som alle PC-er.

c) Målet med et samlebånd er å utføre instruksjoner i parallell

d) Samlebånd gjør at prosessoren gjør ferdig flere instruksjoner per klokke

5) Hva er riktig om samlebånd?

a) Klokkefrekvensen til en prosessor med samlebånd er begrenset av summen av tiden samlebåndet bruker i de ulike trinnene

b) Klokkefrekvensen til en prosessor med samlebånd er begrenset av det tregeste trinnet

c) Klokkefrekvensen til en prosessor med samlebånd er begrenset av gjennomsnittstiden til trinnene

d) En prosessor har ulik klokkefrekvens for de ulike trinnene

6) Dersom du har et vanlig (in-order execution) samlebånd, hvilke dataavhengigheter vil skape problemer i følgende program (Dark-syntaks – se vedlegg)?

```
SUB $8, $3, $1
ADD $3, $1, $8
```

a) Avhengigheten på grunn av register 1

b) Avhengigheten på grunn av register 3

c) Avhengigheten på grunn av register 8

d) Det finnes ingen avhengighet

7) Hva skiller indirekte adresseringsmodus fra direkte adresseringsmodus (velg det alternativet som er mest riktig)?

a) Direkte er enklere enn indirekte adressering fordi indirekte adressering krever et ekstra oppslag

b) Indirekte adressering går ofte tregt og er gammeldags, og moderne prosessorer benytter kun direkte adressering

c) Indirekte adressering adresserer relativt til stakkregisteret, mens direkte adressering adresserer relativt til instruksjonsregisteret

d) Indirekte adressering adresserer relativt til stakkregisteret, mens direkte adressering adresserer relativt til statusregisteret

8) Hvilket felt er vanlig å ha i instruksjonsordet?

a) Funksjonskode

b) Statusregister

c) Adressen instruksjonsordet ligger i

d) Adressen til stakkregisteret

9) Hvilken prosessortype har som oftest lengst instruksjonsord?

a) Akkumulatormaskin

b) Load/store-maskin

c) Stakkmaskin

10) Hvilket utsagn om EEPROM (Electronically Erasable Read Only Memory) er FEIL?

a) Innholdet kan slettes ved hjelp av spesielle elektriske pulser like raskt som lesing

b) EEPROM-brikken kan skrives om mange ganger.

c) Innholdet slettes byte for byte

d) Det er dyrt og går tregt å skrive/slette.

11) Hva er fordelen med et asynkron sammenkobling av to datamaskiner sammenlignet med en synkron sammenkobling?

a) Det er mye raskere

b) Det tøler støy bedre

c) Datamaskinene kan være mer uavhengige og trenger ikke kjøre på samme klokke

d) Det finnes ingen fordeler

12) Hvilken av følgende påstander om PCI-bussen er mest FEIL?

a) PCI bussen er seriell

b) PCI bussen er parallell

c) PCI bussen er egnet til å koble sammen bl.a. lydkort i en pc

d) PCI er en forkortelse for Peripheral Component Interconect

13) Hva er daisy chain?

a) En måte å sette datamaskiner i nett slik at trafikken flyter bra mellom maskinene

b) En måte å organisere disker på for å få høyere båndbredde

c) En teknikk som benyttes i hurtiglager design for å få større treff rate

d) Benyttes bl.a. i avbruddshåndtering for å gi prioritet til riktig avbrudd

14) Hvilket av følgende utsagn er mest FEIL om harddisker?

a) Harddisker glemmer innholdet sitt etter en gitt periode

b) Harddisker er mekaniske og tåler dårlig støt

c) Harddisker kan bare fysisk skrives på en gang (men logisk mange ganger)

d) Harddisker har roterende plater, og man må vente til den har rotert til riktig plass før man kan lese data

15) Hvilket av følgende utsagn er mest feil om dynamisk lager (DRAM)?

a) DRAM er billigere enn SRAM (statisk lager)

b) DRAM lagrer dataene låst i porter

c) DRAM trenger oppfriskning

d) DRAM benyttes i stort sett alle PCer

16) Hvilket av følgende utsagn er mest riktig om dynamisk lager (DRAM)?

a) DRAM er mest brukt som lagereknologi i hurtiglager

b) DRAM har ofte feil, men man kan benytte selvkorrigerende kode for å skjule disse feilene

c) DRAM har blitt mye raskere de siste 10 årene

d) Det finnes ingen standarder for hvordan DRAM skal pakkes inn i moduler

Oppgave 2 – Definisjoner (14%)

Velg et ord som best fullfører setningen. Ikke bruk samme ordet mer enn én gang. Du vil ikke få bruk for alle ordene. Svar med setningsnummer og ord, f.eks. 1a,2b,3c,4d osv. Hvert riktig svar gir 2%.

a) Unntak (eng. exception)
b) Tilstandsmaskin (eng. finite state machine)
c) Uhåndtert avhengighet (eng. hazard)
d) Supersamlebånd (eng. superpipelining)
e) Foroverkobling(eng. forwarding)
f) Avbrudd (eng. interrupt)
g) Opphold (eng. stall)
h) Kombinatorisk
i) Forsinkede instruksjoner (eng. delay slot)

1. Bruk av ____ reduserer antall ubrukte klokkesykler p.g.a. overføring av kontroll.
2. En uventet hendelse som skjer inni prosessorer som medfører overføring av kontroll er _____.
3. _____ betyr at utganger bare er avhengig av nåværende innganger.
4- _____ er en teknikk for å aksessere data i et samlebånd før data er brukt til å oppdatere tilstanden til et program.
5. _____ er et sett med tilstander med definerte utganger som skifter etter regler basert på inngangsverdier
6. _____ er utstedt av prosessoren og er en dynamisk ekvivalens med nop (no operation) instruksjonen
7. _____ er en situasjon i et samlebånd hvor samlebåndet ikke kan forstette å fullføre en instruksjon hver klokkesyklus uten å medføre en ukorrekt oppdatering av programmet.

Oppgave 3 – Kortsvarsoppgaver (24%)

Noen av oppgavene krever et eksakt tallsvar eller uttrykk. Andre krever en tekstlig forklaring. Svar kort, oversiktlig og strukturert på oppgavene. Strek gjerne under sentrale begreper som omtales. Hver oppgave er 2%.

1) Hva er forutsigelse av forgreininger (eng. branch prediction)? Hva er bakgrunnen til at man trenger dette?
2) Hva er hensikten med avbrudd?

3) Hvordan er normalt forholdet mellom overføringsmengde mellom CPU og hurtiglager og hurtiglager og hovedlager?

4) Hvilke egenskaper har DRAM som gjør at man benytter det i PCer?

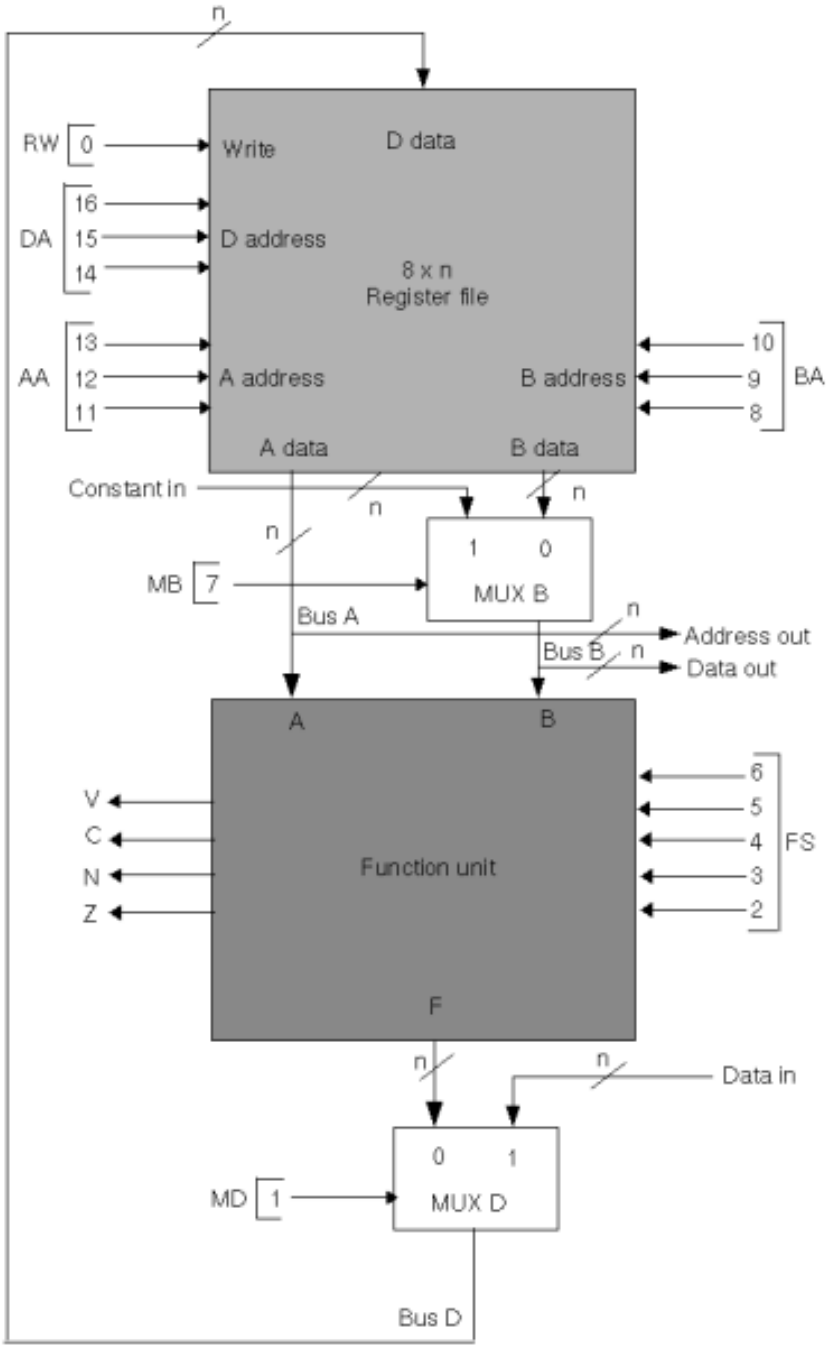
5) Hvorfor finnes konseptet RAID?

6) Hva er Direct Memory Access (DMA) og hvilke fordeler har metoden sammenlignet med andre I/O-metoder?

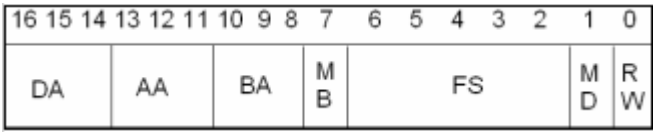
7) Ved representasjon av flyttall er det et begrep som heter ”biased reprsentation” eller forskjøvet representasjon. Det består i av at en konstant legges/subtraheres fra eksponenten. Hvorfor gjør man dette?

.

8) Prosessorer har støtte for at en prosedyre kan kalle en annen prosedyre. Hvordan vet systemet hvor det skal fortsette når den kalte prosedyren er ferdig? Forklar én metode, og legg ved et kort kommentert kodeeksempel.



DA, AA, BA		MB		FS		MD		RW	
Function	Code	Function	Code	Function	Code	Function	Code	Function	Code
R0	000	Register	0	*F* = A	00000	Function	0	No Write	0
R1	001	Constant	1	*F* = A + 1	00001	Data In	1	Write	1
R2	010			*F* = A + B	00010				
R3	011			*F* = A + B + 1	00011				
R4	100			*F* = A + ~B	00100				
R5	101			*F* = A+ ~B+ 1	00101				
R6	110			*F* = A - 1	00110				
R7	111			*F* = ~A	00111				
				F = A ^ B	01000				
				F = A v B	01010				
				F = A XOR B	01100				
				F = A	01110				
				F = sr A	10000				
				F = sl A	10001				



Bruk tabellene gitt ovenfor, som beskriver styreordet til en prosessor, og figuren på forrige side, som skisserer en utførende enhet i denne prosessoren, til å besvare oppgavene under.

9) Hva blir styreordet for operasjonen `R5 <- R2` der x skal beskrive ubrukte bits?

10) Hvilken mikrooperasjon tilsvarer styreordet `110010xxxx0000101` der x beskriver ubrukte bits?

For en gitt CISC-prosessor kan MOV-instruksjonen i assembler være definert med flere adresseringsmodi slik:

```
MOV  Rn, #Imm      ; Immediate
MOV  Rn, Addr      ; Direkte
MOV  Rn, [Addr]    ; Indirekte
MOV  Rn, Rm        ; Register
MOV  Rn, [Rm]      ; Register indirekte
MOV  Rn, Addr(Rm)  ; Displacement
PUSH Rn            ; Stakk push
POP  Rn            ; Stakk pop
```

Rn/Rm indikerer register nummer n/m. Addr og Imm kan være tallverdier eller variabler.

MOV fungerer slik: MOV til-operand, fra-operand

Gitt et lite utdrag av en datamaskins lager på et gitt stadium som skal benyttes for å besvare spørsmålene under:

Hovedlager

Adresse	Verdi
0	71
1	54
2	12
3	86
4	28
5	56
6	3
7	8
8	13
9	7
10	45
11	82

Registre

Registernr	Verdi
R0	5
R1	15
R2	8
R3	7
R4	8
R5	55
R6	9
R7	35

Definerte variabler

Navn	Verdi
teller	64
i	3
antall	4
tab	3
x	7
retur	11

11) Hva vil verdien til R1 være etter følgende instruksjon: `MOV R1, [R2]`?

12) Hva vil verdiene til `R4,R5,R6 og R7` være etter følgende instruksjoner:

```
PUSH R4
MOV  R5, #antall
MOV  R7, [R5]
POP   R5
MOV  R6, 1(R5)
MOV R7, 10
```

Oppgave 4 – Assemblerprogrammering (30%)

Hver oppgave er 15%.

1) Skriv et assembler program i Dark load/store maskin (se vedlegg) som gjør følgende jobb:

```
for(i=0; i≤n; i++) a[i]=a[i]+1
```

Anta at:

`$1` inneholder adressen for a[0], første elementet av en tabell med n+1 ord

`$2` inneholder verdien n.

2)

Skriv et Dark load/store maskin (se vedlegg) programkode som implementerer en pseudoinstruksjonen:

`bodd $2, LOC`

Denne instruksjonen fører til et hopp til LOC hvis tallet i $o er et oddetall.

Hvis du ikke skjønte oppgaven helt: Se for deg at en gammel prosessor av Dark med en load/store arkitektur som hadde en instruksjon som nå ikke finnes. Denne instruksjonen het *bodd*, og fungerer som beskrevet over. Du skal gjøre det mulig å kjøre programmer som er skrevet for den gamle arkitekturen på den nye arkitekturen ved å bytte ut den gamle instruksjonen med en eller flere eksisterende instruksjoner.

Skriv ned forutsetninger du tar for å løse oppgaven.

Svarark for oppgave 1

Studentnr: _____

Fagnummer: _____

Eksamensdato: _____

Side _____ av _____

	a	b	c	d
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				

NB! Ikke glem å levere dette arket! Bla igjennom den papirbunken du gir fra deg til slutt, for å sjekke at svar på avkrysningsoppgaven er med.

VEDLEGG – Dark load/store maskin

Syntax	Semantik
`add` REG,REG,{REG|NUM|VAR}	reg ← reg + {reg|NUM|VAR}
`and` REG,REG,{REG|NUM|VAR}	reg ← reg ∧ {reg|NUM|VAR}
`band` REG,REG,{REG|NUM|VAR}	reg ← reg .∧ {reg|NUM|VAR}
`bnot` REG,REG	reg ← .¬ reg
`bor` REG,REG,{REG|NUM|VAR}	reg ← reg .∨ {reg|NUM|VAR}
`bxor` REG,REG,{REG|NUM|VAR}	reg ← reg .⊕ {reg|NUM|VAR}
`call` ETIKETT	returstack ← pc pc ← ETIKETT
`dec` REG	reg ← reg - 1
`div` REG,REG,{REG|NUM|VAR}	reg ← reg / {reg|NUM|VAR}
`inc` REG	reg ← reg + 1
`jeq` REG,REG,ETIKETT	{|pc ← ETIKETT}
`jge` REG,REG,ETIKETT	{|pc ← ETIKETT}
`jgt` REG,REG,ETIKETT	{|pc ← ETIKETT}
`jle` REG,REG,ETIKETT	{|pc ← ETIKETT}
`jlt` REG,REG,ETIKETT	{|pc ← ETIKETT}
`jmp` ETIKETT	pc ← ETIKETT
`jne` REG,REG,ETIKETT	{|pc ← ETIKETT}
`load` REG,{REG,NUM|NUM|VAR}	reg ← {mem[reg+NUM]|NUM|VAR}
`mod` REG,REG,{REG|NUM|VAR}	reg ← reg \ {reg|NUM|VAR}
`mul` REG,REG,{REG|NUM|VAR}	reg ← reg * {reg|NUM|VAR}
{`mv`|`mov`} REG,REG	reg ← reg
`not` REG,REG	reg ← ¬ reg
`or` REG,REG,{REG|NUM|VAR}	reg ← reg ∨ {reg|NUM|VAR}
`ret`	pc ← returstack
`store` REG,{REG,NUM|VAR}	{mem[reg+NUM]|VAR} ← reg
`sub` REG,REG,{REG|NUM|VAR}	reg ← reg - {reg|NUM|VAR}
`xor` REG,REG,{REG|NUM|VAR}	reg ← reg ⊕ {reg|NUM|VAR}