NTNU

Fakultet for informasjonsteknologi og elektroteknikk

Mal for sensorveiledning

Emnekode	TDT4160
Emnenavn	Datamaskiner
Emneansvarlig	Magnus Jahre
Eksamensdato	11. desember 2025

Emnets læringsutbyttebeskrivelser angitt i kunnskaper, ferdigheter og generell kompetanse. (Henvisning med lenke til studieplan på NTNUs nettsider er tilstrekkelig)	Se: https://www.ntnu.no/studier/emner/TDT4160 Det er også utarbeidet en mer detaljert læringsutbyttebeskrivelse som er gjort tilgjengelig for studentene på emnets nettside og for sensurkommisjonen.
Pensum	Patterson and Hennessy, **Computer Organization and Design — RISC-V Edition: The Hardware Software Interface (2nd Edition),** Morgan Kaufmann Publishers, 2020, Paperback ISBN: 978-0-12-820331-6, eBook ISBN: 9780128245583 Følgende kapitler i Patterson and Hennessy er pensum: • Kapittel 1.1 til 1.13 • Kapittel 2.1 til 2.12, 2.14, 2.22 og 2.23 • Kapittel 3.1 til 3.6, 3.9 og 3.10 • Kapittel 4.1 til 4.11, 4.15 og 4.16 • Kapittel 5.1 til 5.4, 5.6, 5.7, 5.10, 5.16 og 5.17 • Kapittel 6.1 til 6.8, 6.11, 6.14 og 6.15 • Appendiks A-1 til A-3, A-5, A-7 til A-11 (unntatt beskrivelsene av Verilog-kode) I tillegg er øvingene og forelesningene pensum.
Læringsaktiviteter i emnet	Forelesninger, obligatoriske øvinger, og selvstudium.
Eventuelle formelle krav til besvarelsen	Ingen.
Hvordan de ulike oppgavene i eksamenssettet er vektlagt	Det gis inntil 100 poeng på denne eksamenen. Poengfordeling er mellom oppgavene er definert i Inspera og gjengitt under.

Versjon 1, sist oppdatert 12. desember 2025.

Endringslogg:
- V1: Første versjon.

Karakterskala som er benyttet:

Symbol	Betegnelse	Generell, ikke fagspesifikk beskrivelse av vurderingskriterier
A	Fremragende (89 – 100 poeng)	Fremragende prestasjon som klart utmerker seg. Kandidaten viser svært god vurderingsevne og stor grad av selvstendighet.
B	Meget god (77 – 88 poeng)	Meget god prestasjon. Kandidaten viser meget god vurderingsevne og selvstendighet.
C	God (65 – 76 poeng)	Jevnt god prestasjon som er tilfredsstillende på de fleste områder. Kandidaten viser god vurderingsevne og selvstendighet på de viktigste områdene.
D	Nokså god (53 – 64 poeng)	En akseptabel prestasjon med noen vesentlige mangler. Kandidaten viser en viss grad av vurderingsevne og selvstendighet.
E	Tilstrekkelig (41 – 52 poeng)	Prestasjonen tilfredsstiller minimumskravene, men heller ikke mer. Kandidaten viser liten vurderingsevne og selvstendighet.
F	Ikke bestått (0 – 40 poeng)	Prestasjon som ikke tilfredsstiller de faglige minimumskravene. Kandidaten viser både manglende vurderingsevne og selvstendighet.

1 Instruksjonssett

Instruksjon	Opcode	Funct3	Funct7
add	0110011	000	0000000
lw	0000011	010	—
sll	0110011	001	0000000
xor	0110011	100	0000000
andi	0010011	111	—
sw	0100011	010	—
slt	0110011	010	0000000
sra	0110011	101	0100000
sub	0110011	000	0110000
addi	0010011	000	—

Faglærers anmerkning: På oppgave 1.1, 1.2, og 1.3 er det fort gjort å legge inn verdier som Inspera ikke kjenner igjen, og det må vi korrigere for når vi retter.

I denne oppgaven gir vi 1 poeng for hvert riktige svar. Feil svar/ubesvart gir 0 poeng.

Oppgave 1.1 Angi maskinkode for instruksjonen `slt x10, x15, x20`. (6p)

Svar:
- opcode: 0110011
- rs1: 01111
- rs2: 10100
- rd: 01010
- func3: 010
- func7: 0000000

Oppgave 1.2 Angi maskinkode for instruksjonen `andi x5, x4, 0x7fb` (5p)

Svar:
- opcode: 0010011
- rs1: 00100
- rd: 00101
- imm: 0111 1111 1011
- func3: 111

Her må man konvertere fra heksadesimal til binær. 0x7 er 7 i desimal og 0111 i binær, 0xf er 15 in desimal og 1111 i binær, mens 0xb er 11 i desimal og 1011 i binær. Hver heksadesimale verdi representerer 4 bit, og vi har dermed regnet ut alle de 12 bitene vi trenger til immediate-verdien.

Oppgave 1.3 Angi maskinkode for instruksjonen `sw x30, -16(x14)` (5p)

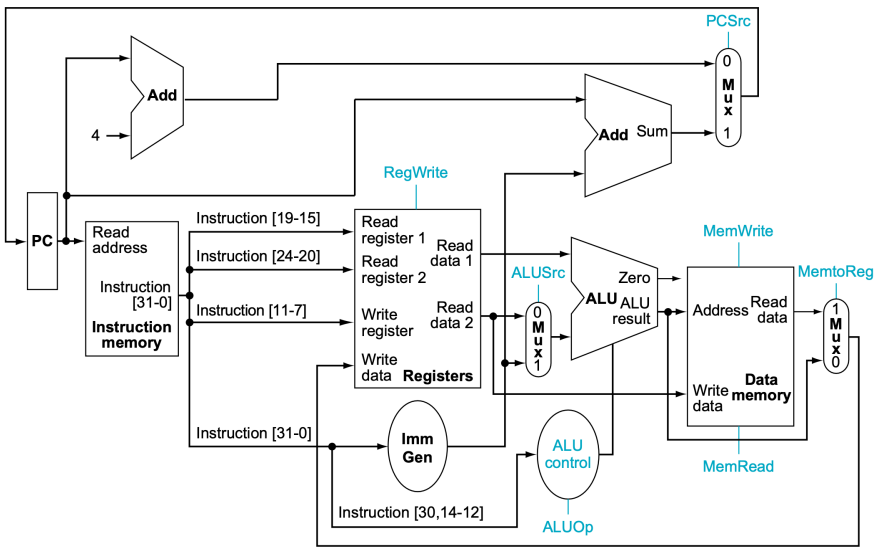
Svar:
- opcode: 0100011
- rs1: 01110
- rs2: 11110
- imm: 1111 1111 0000
- func3: 010

Her må vi konvertere -16 til en binær verdi med 12 bit på 2's komplement form. Det enkleste er starte med at det positive tallet 16 som er 0000 0001 0000 når vi representerer det som et 12-bit binærtall. For å finne -16, tar inverterer vi alle bitene og legger til 1, altså 1111 1110 1111 + 1 som gir 1111 1111 0000. Her kan det være lurt å kontrollregne: $-2^{11} + 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^4 = -2048 + 1024 + 512 + 256 + 128 + 64 + 32 + 16 = -16$.

Totalt blir det gitt inntil 16 poeng på oppgave 1.

2 Enkeltsykelprosessor

Faglærers anmerkning: I denne og påfølgende seksjoner vurderer vi rett svar og rett fremgangsmåte hver for seg. For eksempel vurderer vi rett svar i oppgave 2.1 og rett fremgangsmåte i oppgave 2.2. Det er dermed mulig å få full uttelling på oppgave 2.2 selv om tallene man kommer frem til er feil hvis fremgangsmåten er korrekt.



Denne seksjonen omhandler prosessoren vist over. Du må bruke "don't care" (X) når du kan. I oppgavene som er automatisk rettet, gir vi 0,5 poeng for riktige svar og -0,25 poeng for feil svar. Ubesvarte oppgaver gir 0 poeng.

Faglærers anmerkning: Merk at Inspera i skrivende stund også gir minuspoeng for ubesvarte oppgaver. Hvis dette problemet vedvarer, vil den automatisk utregnede poengsummen i noen tilfeller bli feil. Vi dermed sjekke disse manuelt og om nødvendig overstyre når vi retter. Dette gjelder denne oppgaven og alle påfølgende oppgaver der det gis minuspoeng.

Oppgave 2.1 Prosessoren skal utføre instruksjonen `slt`. Hvilke verdier skal kontrollsignalene under ha? (3p)

Svar:
- RegWrite: 1
- ALUSrc: 0
- PCSrc: 0
- MemWrite: 0
- MemRead: 0
- MemtoReg: 0

Oppgave 2.2 Forklar hvordan kontrollsignalene du oppga i oppgave 2.1 gjør at `slt`-instruksjonen utføres korrekt. (5p)

Svar: `slt` ("set if less than") er en R-type instruksjon, og begge registerverdiene brukes dermed i ALUen (`ALUSrc` = 0). Den påvirker ikke kontrollflyt (`PCSrc` = 0) og hverken skriver eller leser fra minnet (`MemWrite` = 0 og `MemRead` = 0). Utverdien fra ALU må sendes videre til registerfila, og dette får vi til med å sette `MemtoReg` = 0. Vi skal også oppdatere verdien i registerfila, og dette får vi til ved å sette `RegWrite` = 1.

Oppgave 2.3 Prosessoren skal utføre instruksjonen `sb`. Hvilke verdier skal kontrollsignalene under ha? (3p)

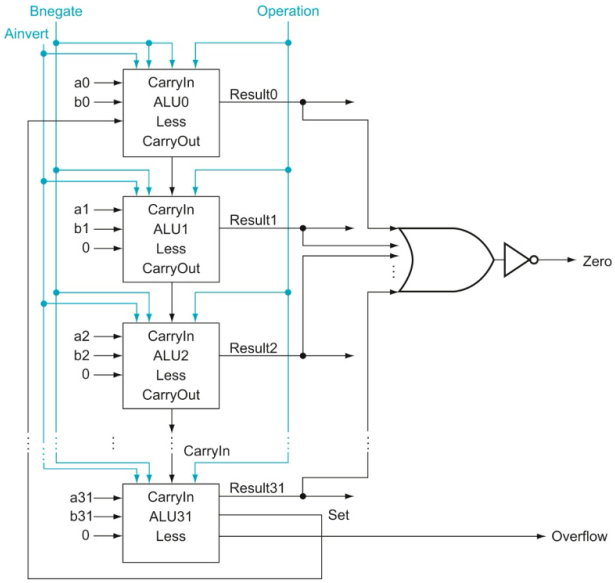
Svar:
- RegWrite: 0
- ALUSrc: 1
- PCSrc: 0
- MemWrite: 1
- MemRead: 0
- MemtoReg: X

Oppgave 2.4 Forklar hvordan kontrollsignalene du oppga i oppgave 2.3 gjør at `sb`-instruksjonen utføres korrekt. (5p)

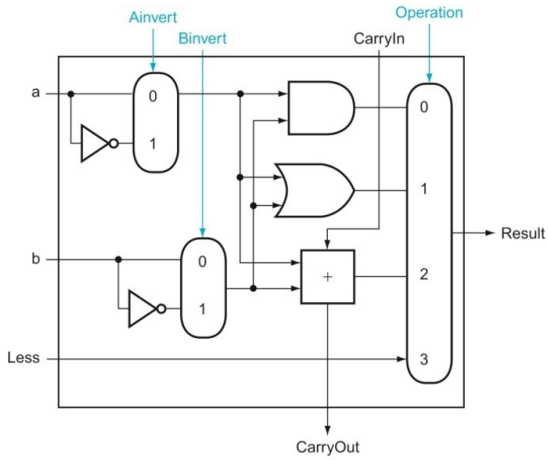
Svar: `sb` ("store byte") er en S-type instruksjon. Her brukes ALUen til å regne ut minneadressen ved å legge verdien lest fra `rs1`-feltet i instruksjonen til immediate-verdien i instruksjonen. Dermed må `ALUSrc` settes til 1. Vi skal skrive til minnet (`MemWrite` = 1), men ikke lese fra minnet (`MemRead` = 0). Vi skal ikke skrive til registerfila (`RegWrite` = 0), og dermed bryr vi oss ikke om hva som ligger på inngangen til registerfila (`MemtoReg` = X). Instruksjonen påvirker ikke kontrollflyt og `PCSrc` settes dermed til 0.

Totalt blir det gitt inntil 16 poeng på oppgave 2.

3 Aritmetisk-logisk enhet



32-bit ALU



1-bit ALU

Denne oppgaven omhandler den 32-bit brede aritmetisk-logiske enheten (ALUen) vist over (til venstre) som igjen er bygget opp av 1-bit ALUene vist over til høyre. `ALU31` inneholder også logikk for å detektere overflyt, men dette er ikke vist i figuren til høyre.

Oppgave 3.1 ALUen skal utføre operasjonen $a - b$. Angi korrekte kontrollsignaler. (3p)

Angi signalverdier i binær med rett antall bit. Vi gir 1 poeng for riktige svar og -0,25 poeng for feil svar. Ubesvarte oppgaver gir 0 poeng.

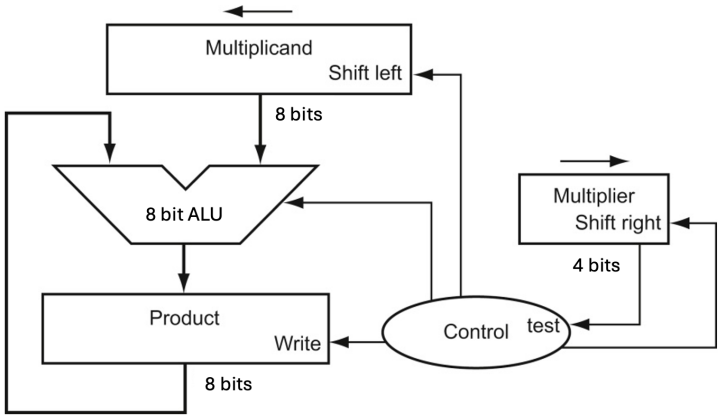
Svar:
- Bnegate: 1
- Ainvert: 0
- Operation: 10

Oppgave 3.2 Forklar hvordan kontrollsignalene du har oppgitt i oppgave 3.1 fører til at operasjonen $a - b$ utføres. (4p)

Svar: Vi realiserer subtraksjon gjennom å representere b på 2's komplement form og så utføre $a + b$. For å få til dette må vi invertere alle bitene i b og legge til en, og det får vi til gjennom å sette `Bnegate` = 1. Vi vil ikke endre på a (`Ainvert` = 0), og så velger vi utgangen fra addisjonsenheten ved å sette `Operation` til 10.

Totalt blir det gitt inntil 7 poeng på oppgave 3.

4 Multiplikator



Denne seksjonen omhandler multiplikatoren over.

Oppgave 4.1 Multiplikatoren skal gange de binære tallene 1010 og 1001 med hverandre, altså utføre operasjonen 1010×1001. Angi verdien som `Product`-registeret inneholder etter hver iterasjon. (4p)

Angi registerverdiene i binær med rett antall bit. Vi gir 1 poeng for hvert riktige svar. Feil svar/ubesvart gir 0 poeng.

Svar: Denne oppgaven er automatisk rettet. Se oppgave 4.2 for utledning.

Faglærers anmerkning: Her kan man i prinsippet bytte rekkefølgen på faktorene og få andre registerverdier. Det er strengt tatt feil, men vi skal ikke trekke for at man blander multiplikand og multiplikator hvis det er den eneste feilen. Dermed må vi ta høyde for dette når vi retter.

Oppgave 4.2 Forklar hvordan du kom frem til svaret i oppgave 4.1. (4p)

Svar: Oppgaven er å regne ut $10 \times 9 = 90$.

Multiplikatoren utfører multiplikasjon omtrent slik som vi gjør den på papir:

					1	0	1	0	$\times$	1001
(i)		0	0	0	0	1	0	1	0	
		0	0	0	0	0	0	0	0	
		0	0	0	0	0	0	0	0	
(ii)	+	0	1	0	1	0	0	0	0	
	=	0	1	0	1	1	0	1	0	

Den eneste forskjellen er at multiplikatoren akkumulerer fortløpende i `Product`-registeret mens vi i eksemplet over summerer alle delverdiene til slutt. Dermed blir verdiene i `Product`-registeret:

- Initialverdi i `Product`: 0000 0000.
- Verdien i `Product` etter første iterasjon: 0000 0000 + 0000 1010 = 0000 1010; se linje (i) i utregningen.
- Verdien i `Product` etter andre iterasjon: 0000 1010 + 0000 0000 = 0000 1010.
- Verdien i `Product` etter tredje iterasjon: 0000 1010 + 0000 0000 = 0000 1010.
- Verdien i `Product` etter fjerde iterasjon: 0000 1010 + 0101 0000 = 0101 1010; se linje (ii) i utregningen.

Igjen kan det lønne seg å kontrollregne: $0101\,1010 = 2^6 + 2^4 + 2^3 + 2^1 = 64 + 16 + 8 + 2 = 90$

Totalt blir det gitt inntil 8 poeng på oppgave 4.

5 Flyttall

$$(-1)^s \cdot (1+\text{Fraction}) \cdot 2^{(\text{Exponent}-127)}$$

Denne seksjonen omhandler flyttall som følger formatet definert over.

Oppgave 5.1 Hvilket desimaltall representerer flyttallet `0x40980000`? Feil svar/ubesvart gir 0 poeng. (2p)

Svar: Denne oppgaven er automatisk rettet. Se oppgaven under for utledning.

Faglærers anmerkning: Her må vi igjen passe på at Inspera ikke trekker poeng på grunn av måten svaret er oppgitt på. Det har videre kommet meldinger om at Inspera ikke alltid lykkes med å lagre tallet som skrives inn. Hvis kandidaten har oppgitt rett tallsvar på oppgave 5.2, skal det dermed gis 2 poeng på denne oppgaven også.

Oppgave 5.2 Forklar hvordan du kom frem til svaret i oppgave 5.1. (6p)

Svar: Vi begynner med å oversette hvert siffer i det heksadesimale tallet til binær: 0x4 gir 0100, 0x0 gir 0000, 0x9 gir 1001 og 0x8 gir 1000. Så fyller vi ut flyttallsformatet:

sign	exponent	fraction
31	30 23	22 0
0	100 0000 1	001 1000 0000 0000 0000 0000

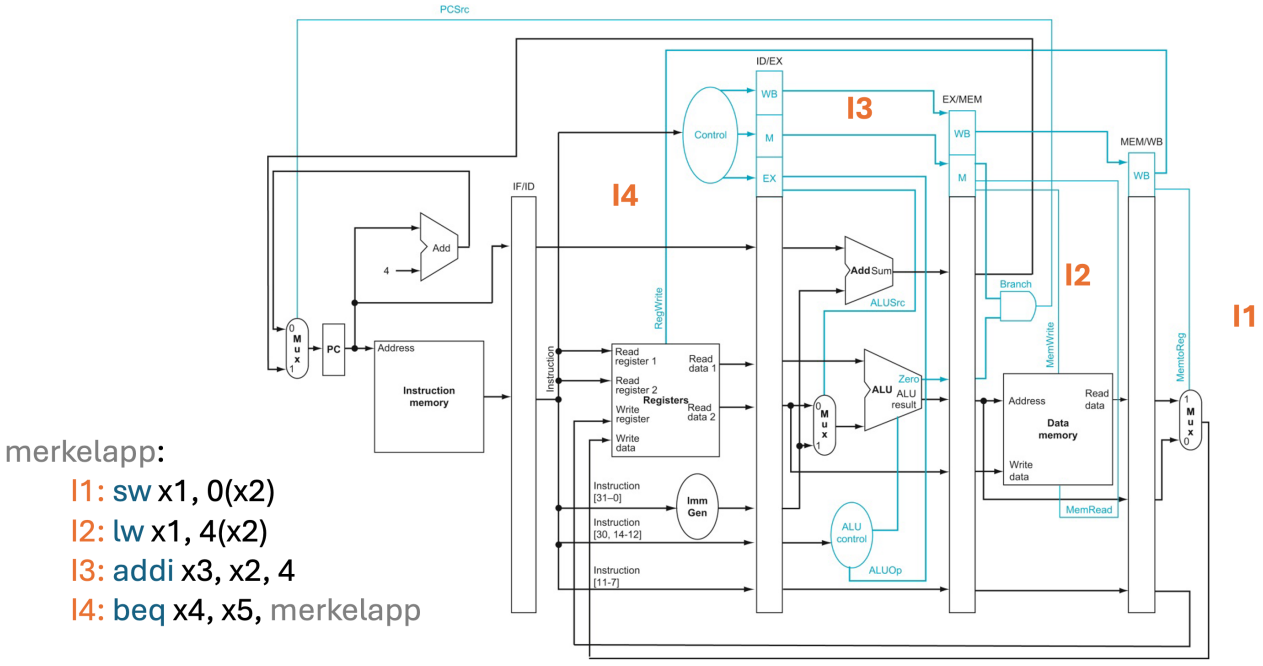
Med dette på plass, bruker vi den oppgitte formelen til å finne verdien:

- Siden `s` er 0, vet vi at tallet er positivt; $(-1)^0 = 1$.
- `exponent` er 129 ($2^8 + 2^0$), og dette betyr at vi må gange med $2^{129-127} = 2^2 = 4$.
- `fraction` er $2^{-3} + 2^{-4} = \frac{1}{2^3} + \frac{1}{2^4} = 0,125 + 0,0625 = 0,1875$.

Verdien er altså 4,75 fordi $1 \times (1 + 0,1875) \times 4 = 4,75$.

Totalt blir det gitt inntil 8 poeng på oppgave 5.

6 Samlebåndsprosessor



Oppgave 6.1 Figuren over viser en spesifikk klokkesykel i utføringen av et program på en samlebåndsprosessor. Hva er verdiene til følgende kontrollsignaler i denne klokkesykelen? (5p)

Du må bruke "don't care" (X) når du kan. Vi gir 0,5 poeng for riktige svar og -0,25 poeng for feil svar. Ubesvarte oppgaver gir 0 poeng.

Svar:

- `ID/EX ALUSrc` = 1
- `ID/EX MemWrite` = 0

- `ID/EX MemRead` = 0
- `ID/EX RegWrite` = 1
- `EX/MEM MemWrite` = 0
- `EX/MEM MemRead` = 1
- `EX/MEM RegWrite` = 1
- `EX/MEM MemtoReg` = 1
- `MEM/WB RegWrite` = 0
- `MEM/WB MemtoReg` = X

Faglærers anmerkning: Her er det en viss fare å bomme med en klokkesykel men ellers tenke rett. I dette tilfellet gir vi inntil 3 poeng på denne oppgaven, forutsatt at forklaringen i oppgave 6.2 klart viser at kandidaten har forstått hvordan maskinen virker.

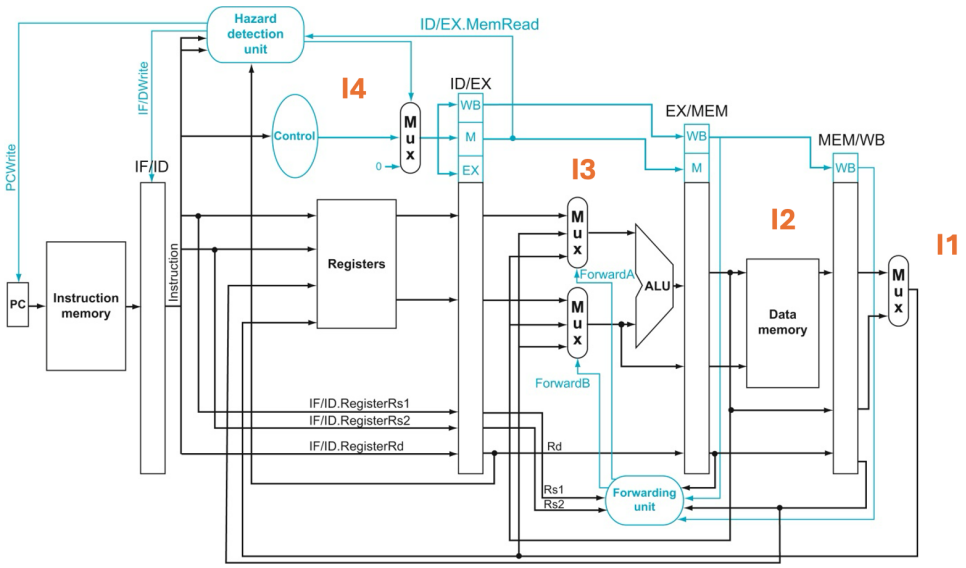
Oppgave 6.2 Hvordan sørger kontrollsignalene for at programmet i oppgave 6.1 utføres korrekt? Forklaringen skal dekke alle kontrollsignaler unntatt `ALUOp` for instruksjonene I1, I2, og I3. (6p)

Svar: Instruksjon I1 er en `sw`-instruksjon, og denne skal ikke gjøre noe i `WB`-steget. Dette oppnår vi med å sette `RegWrite` til 0, og da skal vi også sette `MemtoReg` til X ("don't care").

Instruksjon I2 er en `lw`-instruksjon, og den er i `MEM`-steget i denne klokkesykelen. Den skal dermed lese fra minnet (`MemRead` = 1) i denne klokkesykelen, men ikke skrive til minnet (`MemWrite` = 0) eller påvirke kontrollflyt (`Branch` = 0). Den må også ta med seg kontrollsignalene `MemtoReg` = 1 og `RegWrite` = 1 videre slik at verdien den har lest fra minnet blir skrevet til registerfila når den når `WB` i neste klokkesykel.

Instruksjon I3 er en `addi`-instruksjon som er i `EX`-steget i denne klokkesykelen. Den legger sammen verdien i `x2` med immediate-verdien 4 som vi gjør tilgjengelig for ALUen ved å sette `ALUSrc = 1`. I3 skal ikke gjøre noe når den kommer til `MEM`-steget (`Branch` = 0, `MemRead` = 0, og `MemWrite` = 0), men den skal skrive verdien den regnet ut i `EX`-steget til registerfila når den kommer til `WB`-steget (`MemtoReg` = 0 og `RegWrite` = 1). Merk at oppgaven ikke spør om `ALUOp`.

I1: add x5, x1, x2
I2: add x6, x5, x2
I3: add x7, x5, x6
I4: add x8, x7, x2



Oppgave 6.3 Figuren over viser en ny klokkesykel for et annet program på en annen prosessor. Hvordan sørger prosessoren for at instruksjon I3 utføres korrekt? (5p)

Svar: Utfordringen er at instruksjon I3 er i `EX`-steget i denne klokkesykelen, men den har lest foreldede verdier for `x5` — fordi den skrives av instruksjon I1 som kun har kommet til `WB` — og `x6` — som skrives av instruksjon I2 som er i `MEM`-steget. Dette løses av videresendingsenheten ("forwarding unit"). Mer spesifikt sammenlikner den registeradressene som I1 har lest fra med registeradressene som skrives i `MEM` og `WB`. Da oppdager den at verdien I1 har produsert for `x5` skal erstatte den øverste ALU operanden, og dette oppnås gjennom å sette `ForwardA`-signalet slik at denne verdien velges. Tilsvarende bruker den `ForwardB`-signalet til å videresende I2's `x6` verdi fra `MEM`-steget til ALUens nederste inngang.

Totalt blir det gitt inntil 16 poeng på oppgave 6.

7 Hurtigbuffer og virtuelt minne

Oppgave 7.1 Anta et 2-veis settassosiativt hurtigbuffer med en kapasitet på 256 KiB og en blokkstørrelse på 128 byte. Minneadressene er 32 bit lange, og minnet er byte-adresserbart. Hvilke bit i adressen brukes til offset, indeks, og tag? (3p)

Bitposisjonen til den minst signifikante biten er 0. Vi gir 0,5 poeng for rett svar og 0 poeng for feil svar/ubesvart.

Svar: Denne oppgaven rettes automatisk i Inspera. Svarene er oppgitt i neste oppgave.

Oppgave 7.2 Forklar hvordan du kom frem til adresseformatet i oppgave 7.1. (6p)

Svar: Vi starter med å regne ut antallet adressebit i offset-feltet. $128 = 2^7$ og offsetfeltet blir dermed på 7 bit. Vi legger offsetfeltet på adressebitene fra 0 til 6 fordi offset angir adressene til dataelementer inne i en hurtigbufferblokk.

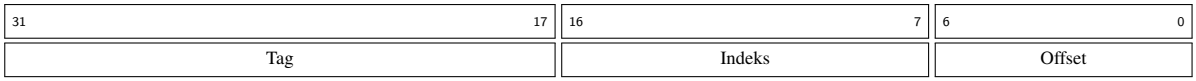
For å regne ut hvor mange bit vi trenger i indeksfeltet, må vi først finne ut hvor mange blokker det er hurtigbufferet:

$$\frac{256\ \text{KiB}}{128\ \text{B}} = \frac{256 \cdot 2^{10}\ \text{B}}{128\ \text{B}} = 2 \cdot 2^{10} = 2048$$

Siden hurtigbufferet er 2-veis settassosiativt, peker hver indeks på to hurtigbufferblokker. Vi har dermed 1024 indekser, og disse legger beslag på 10 adressebit fordi $2^{10} = 1024$. Vi plasserer indeksfeltet på de minst signifikante bitene som fortsatt er tilgjengelig fordi sekvensiell aksess er vanlig, altså bitene fra 7 til 16.

Tag-feltet brukes til å identifisere blokkene vi har lagret i hurtigbufferet, og må derfor inneholde de resterende adressebitene (fra 17 til 31).

Adresseformatet blir dermed:



Enhet	Latens (klokkesykler)	Bomrate
Nivå 1 hurtigbuffer	3	15%
Nivå 2 hurtigbuffer	10	30%
Minnet	75	–

Oppgave 7.3 Hva blir gjennomsnittlig minneaksesstid for systemet beskrevet i tabellen over? (2p)

Svar: Denne oppgaven rettes automatisk i Inspera. Svarene er oppgitt i neste oppgave.

Faglærers anmerkning: Her er det fort gjort å skrive svaret må en måte som Inspera ikke kjenner igjen, og dette må vi passe på når vi retter.

Oppgave 7.4 Forklar hvordan du kom frem til svaret i oppgave 7.3. (4p)

Svar: Andelen forespørsler som treffer i nivå 1 hurtigbufferet er $1 - 0{,}15 = 0{,}85$ og de ser en latens på 3 klokkesykler.

15% av minneforespørslene går videre til nivå 2 hurtigbufferet, og 70% ($1 - 0{,}3 = 0{,}7$) av disse finner dataene de leter etter der. Det vil si at 10,5% ($0{,}15 \cdot 0{,}7 = 0{,}105$) av minneforespørslene ser en latens på 13 ($3 + 10$) klokkesykler.

Forespørslene som bommer i nivå 2 hurtigbufferet ($0{,}15 \cdot 0{,}3 = 0{,}045$; 4.5%) går til minnet og ser en latens på 88 klokkesykler ($3 + 10 + 75$). (Kontrollregning: $0{,}85 + 0{,}105 + 0{,}045 = 1$.)

Den gjennomsnittlige minnelatensen blir dermed:

$$0{,}85 \cdot 3 + 0{,}105 \cdot 13 + 0{,}045 \cdot 88 = 2{,}55 + 1{,}365 + 3{,}96 = 7{,}875$$

Oppgave 7.5 Hva gjør et "Translation Lookaside Buffer (TLB)", og hvorfor er de viktige i datamaskiner som støtter virtuelt minne? (4p)

Svar: En TLB er et lite hurtigbuffer som lagrer oversettelser fra virtuelle til fysiske adresser. TLBen er plassert nært prosessoren, og vi slipper dermed å aksessere sidetabellen i minnet hver gang vi skal oversette fra virtuelle til fysiske minneadresser. Siden programmet bruker virtuelle adresser og minnet bruker fysiske adresser, må vi oversette adresser hver gang vi skal lese eller skrive til minnet. Det vil dermed være ødeleggende for datamaskinens ytelse å bruke mange titalls klokkesykler på å aksessere sidetabellen på hver minneaksess. TLBen gjør at vi unngår dette, og det er grunnen til at den er uunnværlig i systemer som støtter virtuelt minne.

Totalt blir det gitt inntil 19 poeng på oppgave 7.

8 Prosessorer med høyere ytelse og parallelle datamaskiner

Oppgave 8.1 Hva gjør "register renaming" og hvorfor brukes denne teknikken i prosessorer med høy ytelse? (4p)

Svar: "Register renaming" fjerner WAR-farer ("Write After Read") og WAW-farer ("Write After Write") gjennom å tilordne et nytt destinasjonsregister til hver instruksjon. For å kunne få til dette, må maskinen ha (langt) flere fysiske registre enn arkitekturregistre. "Register renaming" er nyttig fordi de senere stegene i samlebåndet da kun trenger å ta hensyn til RAW-farer ("Read After Write"). RAW-farer oppstår når en instruksjon produserer data som en annen instruksjon bruker, og derfor må RAW-farer alltid håndteres.

Faglærers anmerkning: Her gir vi inntil 2 poeng for en god faglig beskrivelse av hvordan "register renaming" fungerer og inntil 2 poeng for en godt faglig begrunnet forklaring av hvorfor den brukes.

Oppgave 8.2 Hva er den viktigste forskjellen på en SISD og en SIMD maskin? Nevn en fordel og en utfordring med SIMD-maskiner. (6p)

Svar: Denne oppgaven omhandler Flynns taksonomi der SISD står for "Single Instruction Single Data" og SIMD for "Single Instruction Multiple Data." Den viktigste forskjellen er dermed at en SISD-maskin utfører én operasjon på ett dataelement, mens en SIMD-maskin utfører én operasjon på flere dataelementer.

En fordel med en SIMD-maskin kontra en SISD-maskin er at man amortiserer kostnaden ved å hente og dekode en instruksjon over flere dataelementer og dermed oppnår høyere effektivitet.

En utfordring med en SIMD-maskiner er å håndtere forgreiningsinstruksjoner. Anta at en SIMD-maskin opererer på fire dataelementer av gangen, altså d_0, d_1, d_2, og d_3, og at operasjonen som skal utføres krever at det legges til 1 på d_0 og d_1 og at det trekkes fra en på d_2 og d_3. I så fall vil maskinen typisk først utføre $d_0 + 1$ og $d_1 + 1$ mens den sørger for at d_2 og d_3 forblir uendret (maskering). Dernest utføres $d_2 - 1$ og $d_3 - 1$ mens d_0 og d_1 holdes uendret.

Faglærers anmerkning: Her gir vi uttelling for alle svar som er godt faglig begrunnet, men alle momenter i spørsmålet må besvares. Det vil si at vi gir inntil 2 poeng for å oppgi en forskjell, inntil 2 poeng for å oppgi en fordel, og inntil 2 poeng for en å oppgi en utfordring.

Totalt blir det gitt inntil 10 poeng på oppgave 8.