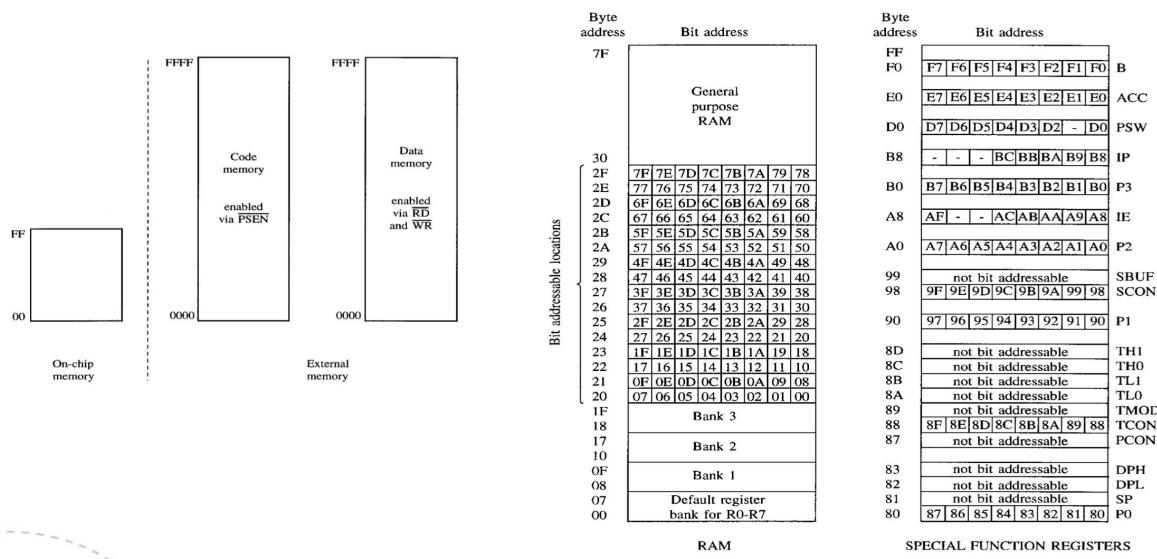


Week 7 adressedekode og adressekart

Minnekart

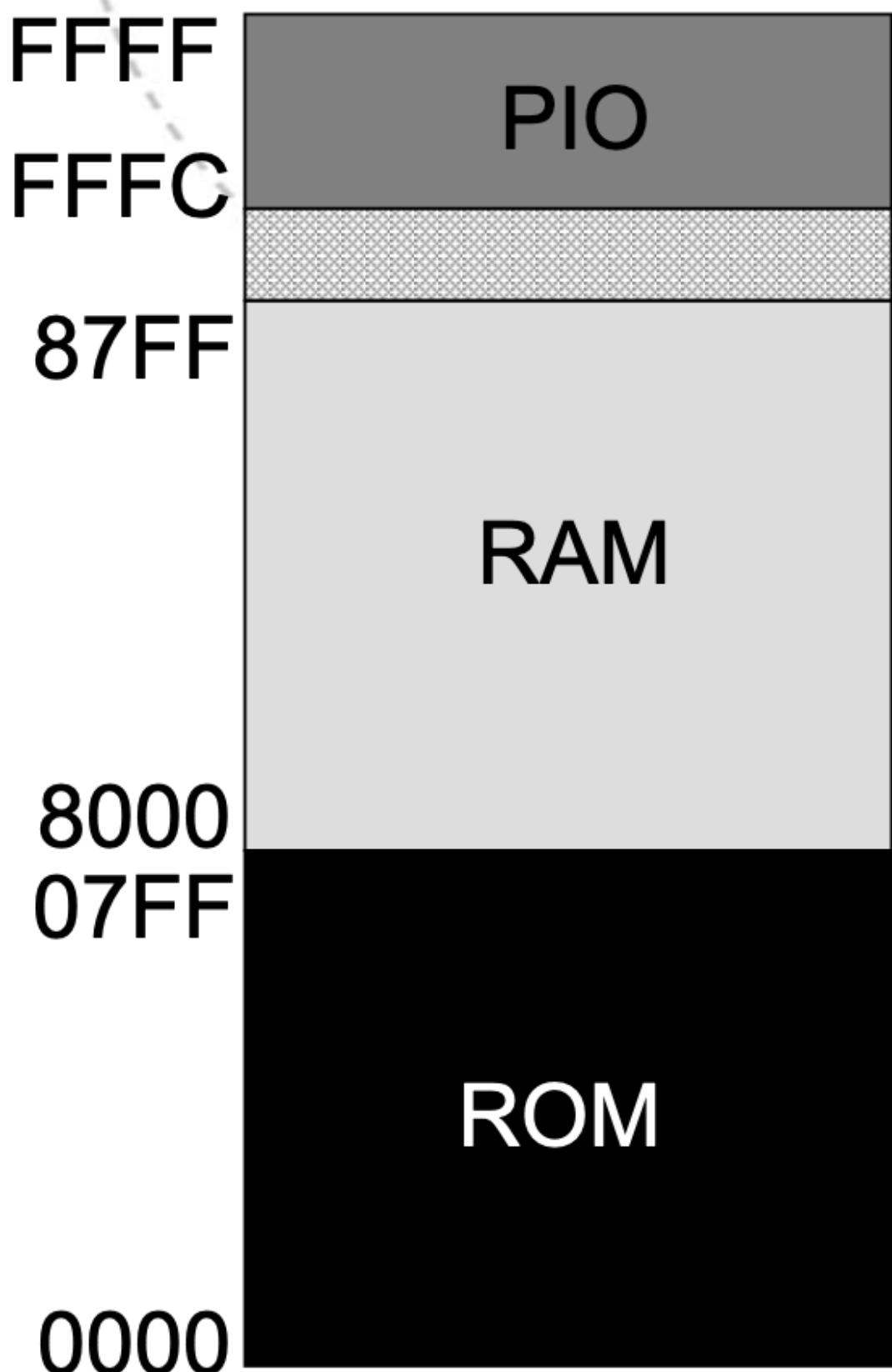
Alt er pekere. Datamaskiner må peke på data eller instruksjon for å skrive eller lese fra dem.



Her har vi minne i prosessoren. Vi har ekstern instruksjonsminne og ekstern dataminne. De to tabellene til høyre er en stor tabell og viser mer detaljert minne i prosessoren.

Fra 30 til 7F har vi fritt minne. Vi har bit-adresserbar minne fra 20 til 20. Alle feltene har sin egen adresse fra 00 til FF.

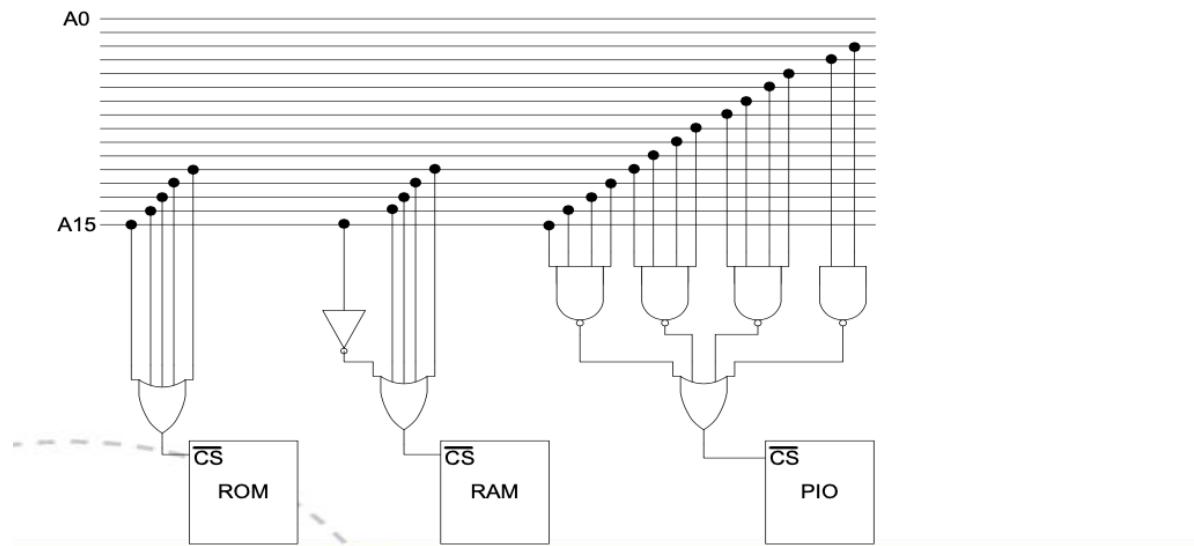
Vi har vanligvis flere enheter som deler bussen. For å kunne snakke med en enhet må vi få tak i den ved å adressere den. Vi har adresserom for alle enhetene. Adressen må dermed dekodes slik at vi faktisk får snakket med den relevante adressen.



Vi bruker et adressekart for å få oversikt over adresserommet til de forskjellige enhetene.

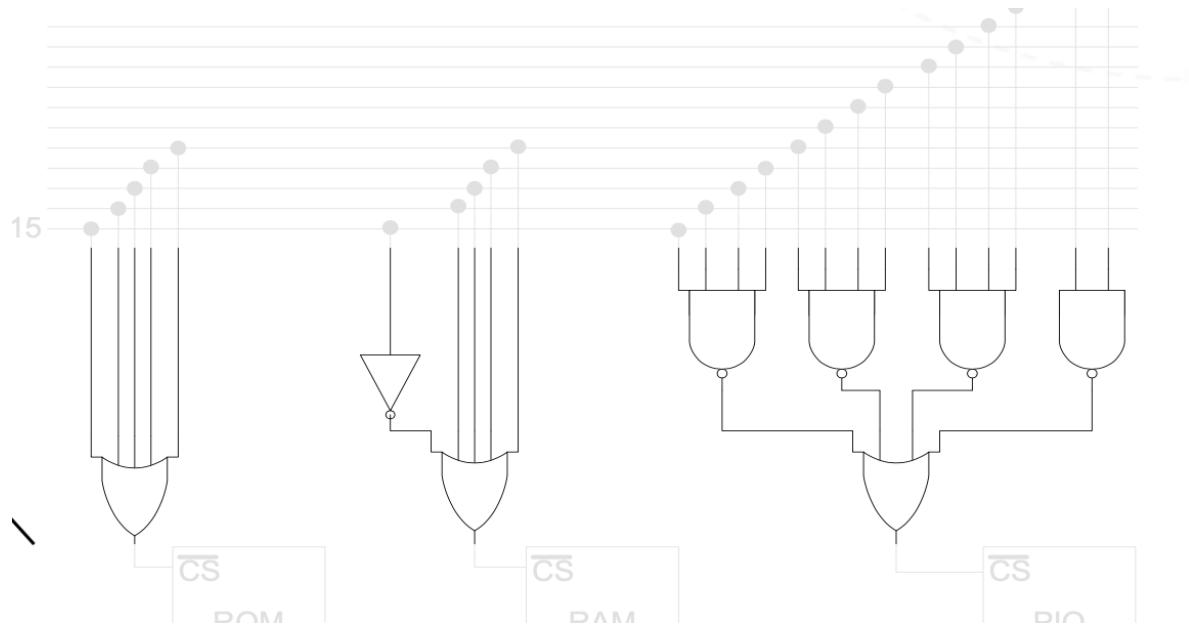
Dersom adressen er mellom 0000 og 07FF så vil ROM bli adressert.

- **Minnekart**
 - Kva ligg på kva minne adresse
 - Kan då aksesere einheitar ved å lese/skrive til minne adresser
- **Eksempel**
 - AdresseBuss minne adresse
 - Dekoding, logikk velg eining
 - Read/write kontrollsignal
 - Dataoverføring databuss



Helt nederst har vi et eksempel. Vi har en adressebuss fra A0 til A15. Dersom adressen er innenfor romet til en enhet vil den enheten bli valgt.

Det må finnes noe logikk som dekoder adressen på adressebussen for å finne adresseområde vi ønsker å lese/skrive til. Dette gjør vi ved å kretser:

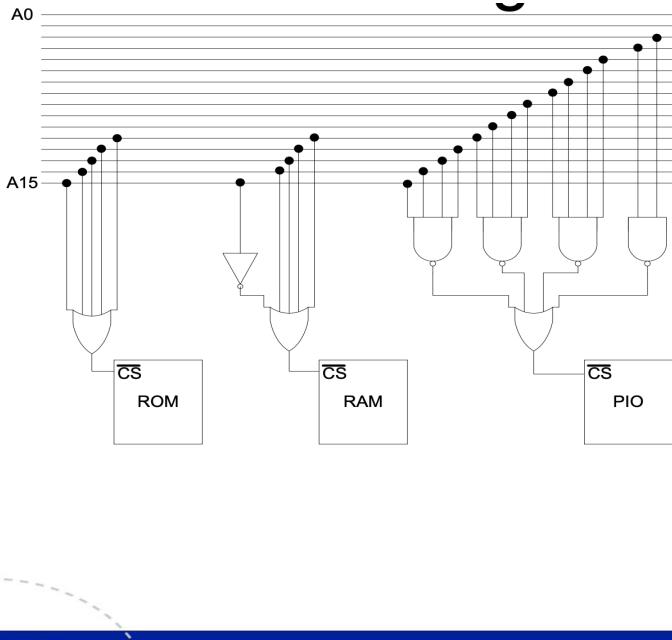


Et sted i RAM-en vår, I/O controller, eller annen enhet, så oppfattes det hvilken adressen som blir lagt ut. Dersom det stemmer med logikken så blir det sendt et aktiveringssignal (CS) til enheten



Vi kan se på eksempelet over. Her ser vi tilkoblingen til ROM-brikken. Øverst har vi adressebussen fra A0 til A15. Vi har signaler A15-A12 som går fra bussen til logikken vår. Dersom CS-signalet er 0 så blir brikken aktivert. Dersom ROM blir aktivert blir dataen lagt ut på databussen.

Adresse dekoding



ROM:

Dekod: 0000 0XXX XXXX XXXX

Høg: 0000 0111 1111 1111

0 7 F F

Låg: 0000 0000 0000 0000

0 0 0 0

RAM:

Dekod: 1000 0XXX XXXX XXXX

Høg: 8 7 F F

Låg: 8 0 0 0

PIO:

Dekod: 1111 1111 1111 11XX

Høg: F F F F

Låg: F F F C

La oss se hva adressen til ROM er. Vi ser at på diagrammet at ROM-en er koblet til de fem øverste bit-ene i adressebussen med en OR port. Alle må være 0 for at ROM skal bli aktivert. Vi skriver X for irrelevante bit. Om vi bytter ut X med 1 så vil vi få den høyeste adressen som dekodingsadressen reagerer på og vi får CS=0. For å få den laveste adressen som dekodingen reagerer på bytter vi ut X med 0 for å få CS=0

La oss se på hva adressen til RAM er. På diagrammet er RAM koblet til A15-A11. Bit A15 er koblet til en inverter. Dette er altså helt lik som ROM, men vi må ha 1 på A15 for å aktivere RAM-en. Vi skriver igjen hvilken bit som må til for å aktivere RAM og skriver X for irrelevante bit. Vi bytter ut X med 1 for få den høyeste mulige verdien for å aktivere og bytter ut X med 0 for å få den laveste mulige verdien for å aktivere.

La oss se på adressen til PIO. Vi trenger 0 på alle inngangene i OR porten. For å få til dette må alle inngangene på NAND portene være 1. Vi bruker ikke kun de to første bitene, ellers må alle bitene være 1. Vi skriver X for de irrelevante bitene. Vi bytter ut X med 1 for å få høyeste mulige adressen og bytter ut X med 0 for å få laveste mulige adresse som fortsatt kan aktivere PIO-en.

Vi kan tegne et adressekart basert på denne informasjonen.