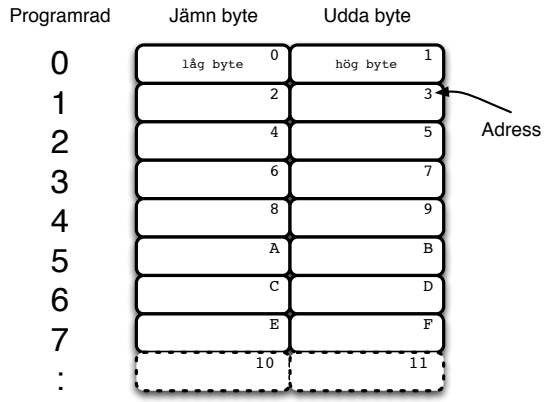
Tabeller i FLASH-minnet

För att använda tabeller i processorns FLASH-minne måste man skilja på

- programrad, och
- minnesadress.

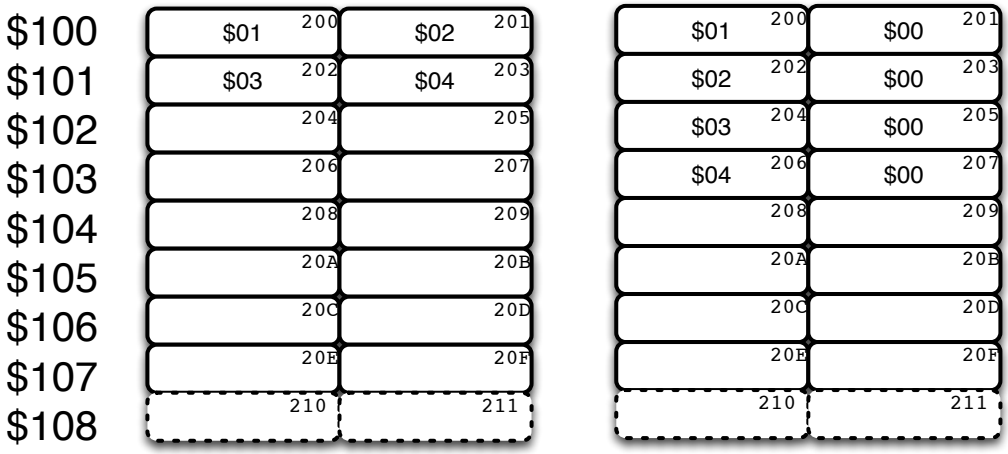
I normalfallet använder vi hela tiden programmets rader när vi skriver program. Det är dessa rader som innehåller programmets instruktioner och det är dessa rader vi hoppar till med `jmp` osv.

En instruktion är 16 bitar bred dvs två bytes. Dessa 16-bitars *word* utgör programraderna och man kan skriva att raden består av en jämn och en udda byte:



Med `.db` och `.dw`, *define byte* respektive *define word*, kan man lägga tabeller i FLASH-minnet (notera att adressen för `TAB` blir `TAB*2`):

```
        .org  $100                  .org  $100
TAB:
        .db $01, $02, $03, $04      .dw $0001, $0002, $0003, $0004
```



Med `.dw` läggs alltid ett word i minnet och det hamnar då på jämna adress automatiskt. Man ser också att det kan vara oekonomiskt för små tabellvärden, då den högre byten sätts till noll.

Instruktionen `lpm` hämtar den byte som `Z`-registret pekar på[1], dvs inte programraden utan adressen. I vänsterfallet ovan kan `lpm` med postinkrement användas för att stega igenom tabellen medan högerfallet kräver ”`adiw ZL,2`” för att peka ut nästa korrekta word.

Detta förklarar också varför en tabell måste innehålla ett jämnt antal bytes. Annars pekar programräknaren inte på hela instruktioner. Kompilatorn ger en varning om detta men lägger nästa instruktion på jämn adress ändå.

[1] `ldi ZH,HIGH(TAB*2)`
`ldi ZL,LOW(TAB*2)`