Sulautetut prosessorijärjestelmät

Mikroprosessorit
-Rakenne ja toiminta
-Esimerkkinä PIC18F452

Määrittely

- Mikroprosessori on ohjelmoitava laite, jolla voidaan suorittaa aritmeettisloogisia operaatiota
- Laitteen yhteydessä jonkunlaista muistia
  - SRAM, ROM, EEPROM, etc.
  - Ei välttämättä samalla sirulla
- Syöttö- ja tulostuslaitteita
  - Näppäimistöt, kytkimet, kiertosäätimet
  - Kirjoittimet, näytöt, LEDit

PIC18F452

- Mutta sehän on ”mikrokontrolleri” eikä mikroprosessori...
- Eroa hankala määritellä tarkasti, yleensä:
  - Mikrokontrollerit ovat pienempiä fyysiseltä kooltaan, sananleveydeltään ja suorituskyvyltään
  - Mikrokontrollereissa usein integroituna oheislaitteita, esim. ajastimia, muuntimia, muistia
  - Mikrokontrollerit vaativat vähemmän tukipiirejä

Perustoiminta

1. Luetaan käsky muistista
2. Dekoodataan käsky
3. Haetaan operandit
4. Suoritetaan operaatio
5. Talletetaan tulokset muistiin
6. Kasvatetaan ohjelmalaskuria (PC)
7. Palataan kohtaan 1.

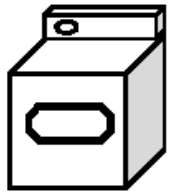
Liukuhihnat
(engl. Pipelines)

- Edellisessä esitetyn menetelmän eri vaiheita voidaan suorittaa rinnakkain
  - Suoritus nopeutuu
  - Laitteen fyysinen kompleksisuus kasvaa hieman
  - Ohjelmoinnissa tai laitteen suunnittelussa huomioitava mahdolliset virhetilanteet
    - Esim. luku ennen kirjoitusta

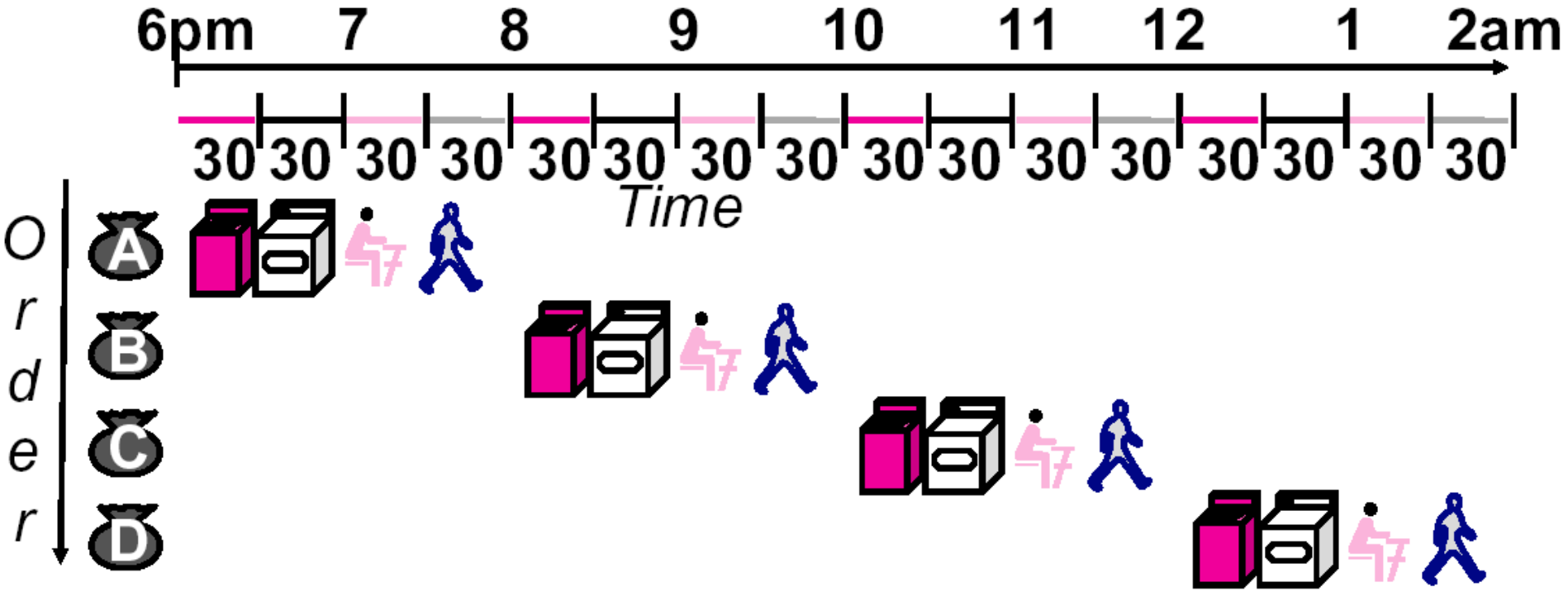
Esimerkki: Pyykinpesu

(By Dan Conners)

- Ann, Brian, Cathy ja Dave pesevät kukin pussillisen pyykkiä
  - Pesu kestää 30 min
  - Kuivaus kestää 30 min
  - Taittelu kestää 30 min
  - Kaappiin asettelu kestää 30 min

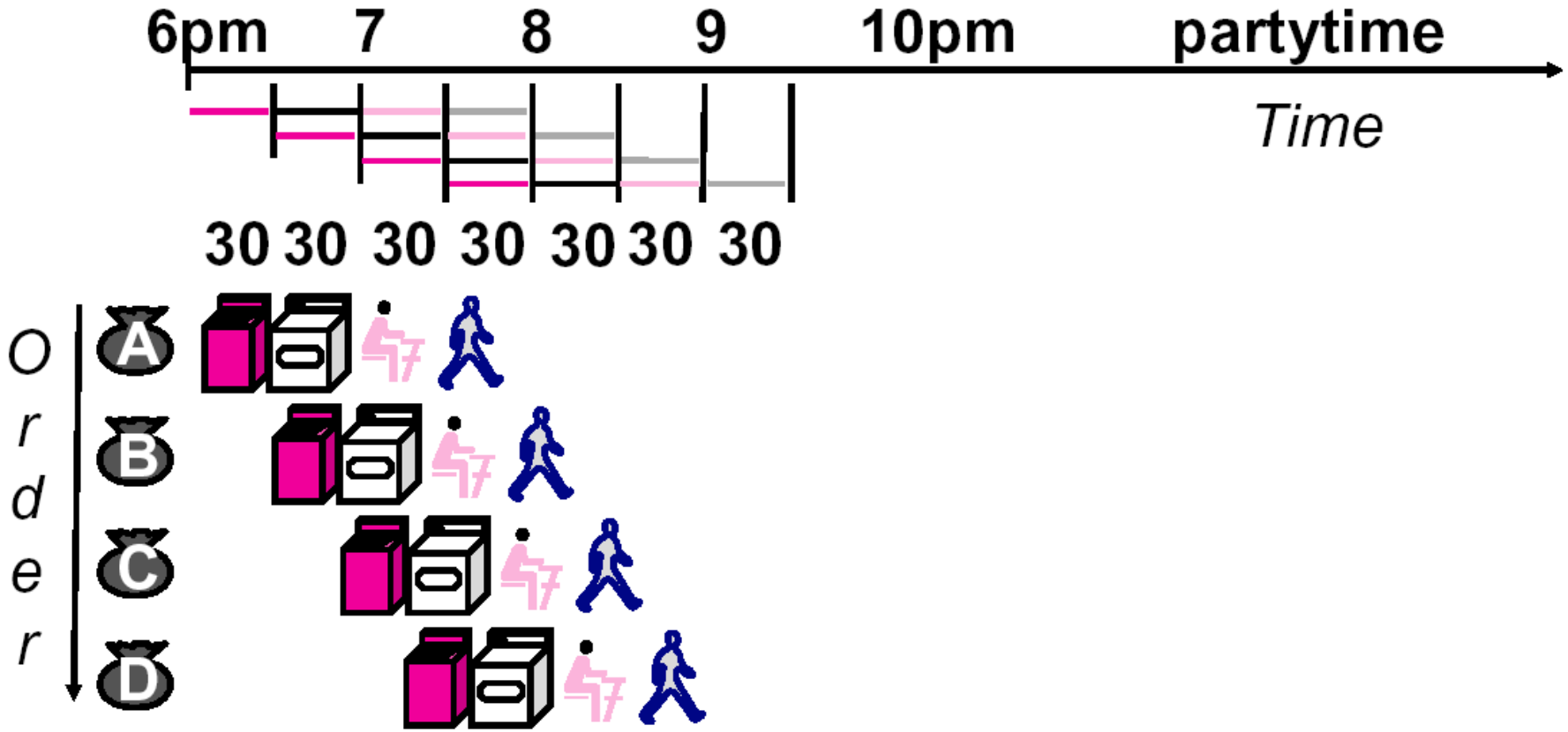


Peräkkäinen menetelmä



- Peräkkäin suoritettuna 4 pussillista saadaan valmiiksi 8 tunnissa
- Kuinka kauan tähän menisi liukuhihnaa käyttäen?

Liukuhihnaa käyttäen



- Liukuhihnaa käyttäen 4 pussin pesuun kuluu vain 3,5 tuntia!

Liukuhihnat
(eng. Pipelines)

- Klassinen RISC arkkitehtuuri:
  1. IF Käskyn nouto muistista
  2. RD Operandien nouto rekistereistä
  3. ALU Operaation suoritus
  4. MEM Datamuistin luku tai kirjoitus
  5. WB Tuloksen tallennus rekisteriin

Liukuhihnat
(eng. Pipelines)

- Intel x86 sarja:
  - Pentium (P5) 5 tasoinen liukuhihna, ~100MHz
  - PentiumPro (P6) 12 tasoa, pohjana Pentium II ja Pentium III malleille, 3 vuodessa (1999-2002) 133 MHz => 1,4 GHz (1:10)
  - Pentium 4 (P7) 28 tasoa, 3,8 GHz
    - 8 ensimmäistä tasoa muuttavat käskyjä yksinkertaisempaan muotoon (CISC => RISC)
    - 20 tasoa varsinaiselle suoritukselle, joista 2 pelkää siirtoviivettä varten
    - Valtava tehonkulutus, 115 W / 119 A @ 3,8 GHz

Liukuhihnan vaaratilanteet
(eng. Pipeline hazards)

- Vaaratilanne sotkee datan joustavan kulun liukuhihnalla
  - Rakenteellinen vaara
  - Datasta johtuva vaara
  - Kontrollista johtuva vaara

Rakenteelliset vaaratilanteet

- Sattuu, kun useampi liukuhihnataso tarvitsee samaa resurssia
- Esim. käskyn haku ja operandien haku molemmat tarvitsevat muistia
- Tätä voidaan vähentää kahdentamalla resursseja
  - Harvard arkkitehtuuri

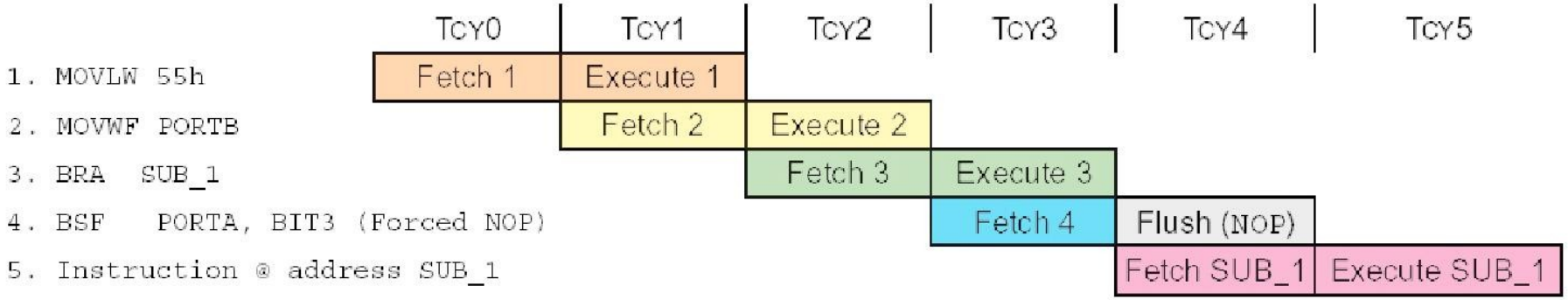
Data vaaratilanteet

- Ohjelma yrittää hakea tietoa ennenkuin aiempi koodisegmentti on ehtinyt kirjoittamaan sen paikoilleen
- Käsky haluaisi operandikseen edellisen käskyn lopputuloksen
- Voidaan ehkäistä:
  - Pysäyttämällä liukuhihna
  - Lisäämällä kanava ALUn ulostulosta sisäänmenoon

Kontrolli vaaratilanteet

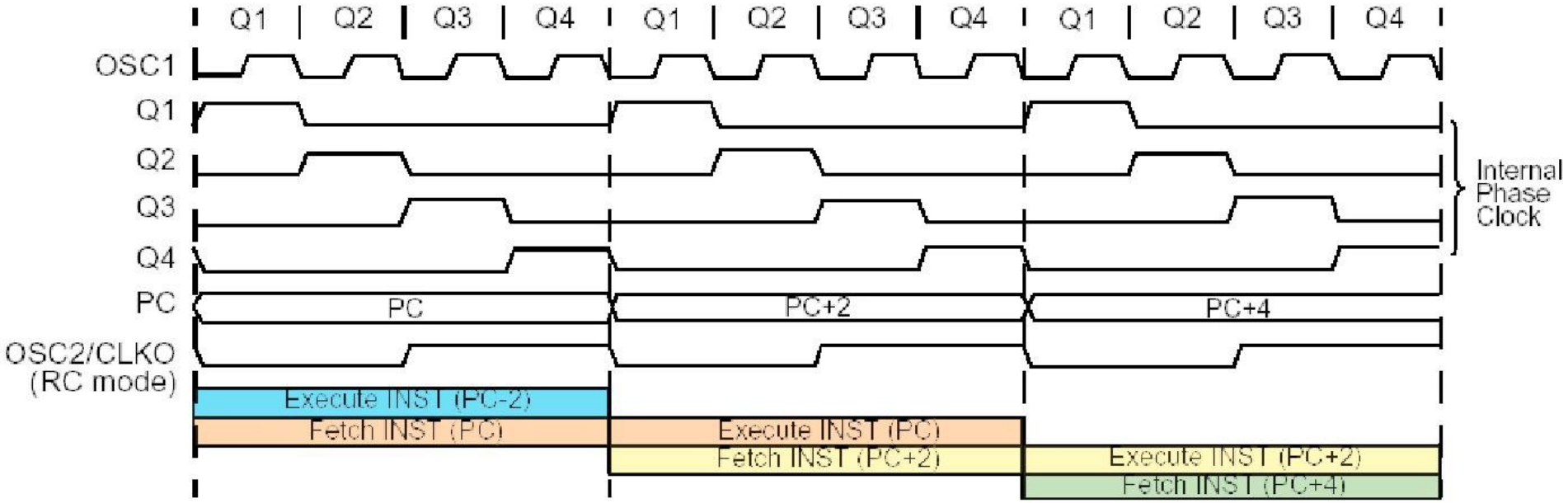
- Ohjelma suorittaa hypyn, jolloin liukuhihnalla perässä tulevat käskyt pitää poistaa
  - Joissain prosessoreissa nämä käskyt suoritetaan aina, ohjelmoija voi sijoittaa näihin kohtiin hyödyllistä laskentaa tai tarvittaessa NOPeja (varsinkin DSP:t, eng. Delay slot)
- Voidaan vähentää:
  - Tunnistamalla hypyt aikaisessa vaiheessa
  - Ennustamalla ehdollisten hyppyjen kohde
  - Varastoimalla hypyjen kohteiden ensimmäiset käskyt
  - Laskemalla uusi PC:n arvo mahdollisimman aikaisin

PIC18F452:n liukuhihna (I)



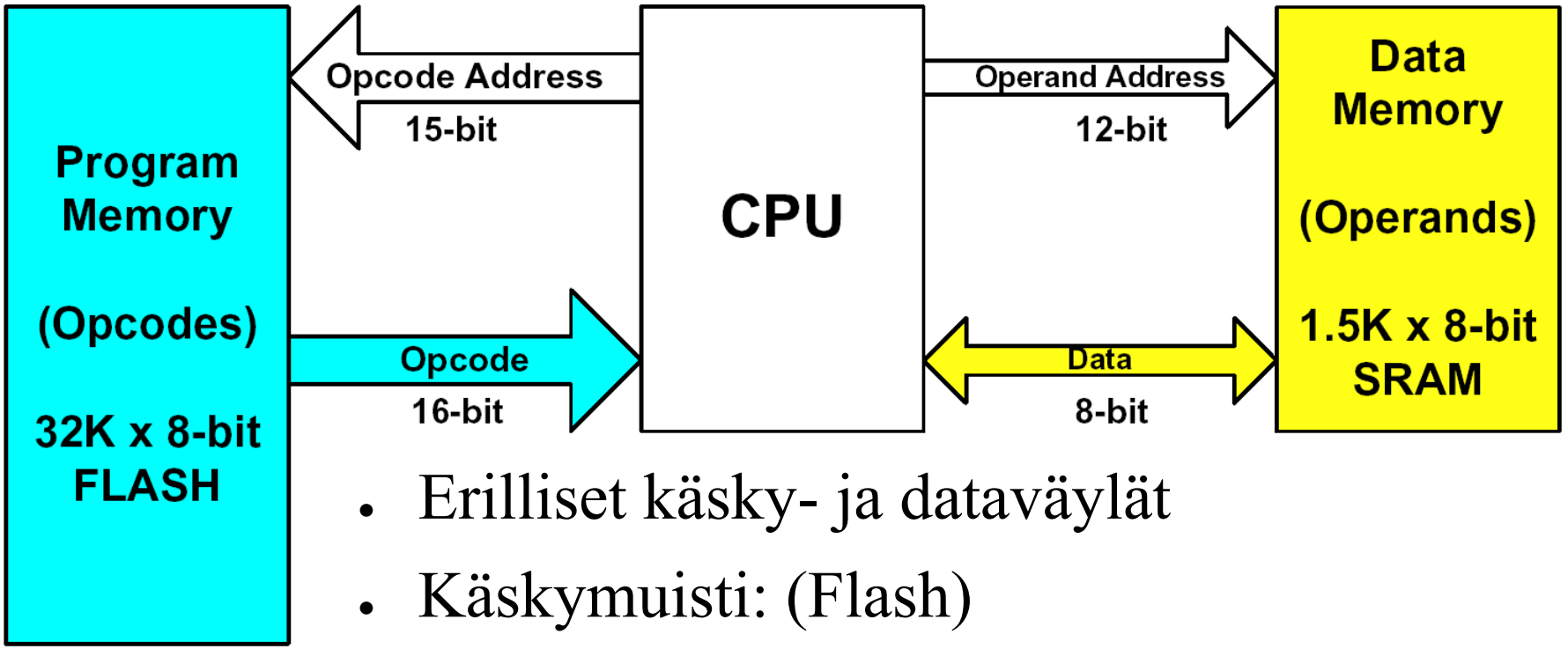
- Kaksi tasoinen
  1. Käskyn haku
  2. Käskyn suoritus
- Poikkeuksena hypyt ja 2 sanaiset käskyt
  - MOVFF, CALL, GOTO ja LFSR

PIC18F452:n liukuhihna (II)



- Molemmat koostuvat 4 kellojaksosta
  - Haku: Q1: kasvattaa PC:tä, Q2: kirjoittaa osoitteen, Q3: hakee käskyn ja Q4: lukitsee käskyn
  - Suoritus: Q1: purkaa käskyn, Q2: hakee operandit, Q3: suorittaa käskyn ja Q4: tallettaa tuloksen
    - Suoritusvaiheen rakenne vaihtelee hieman käskyn mukaan, mutta tämä on tyypillinen sekvenssi

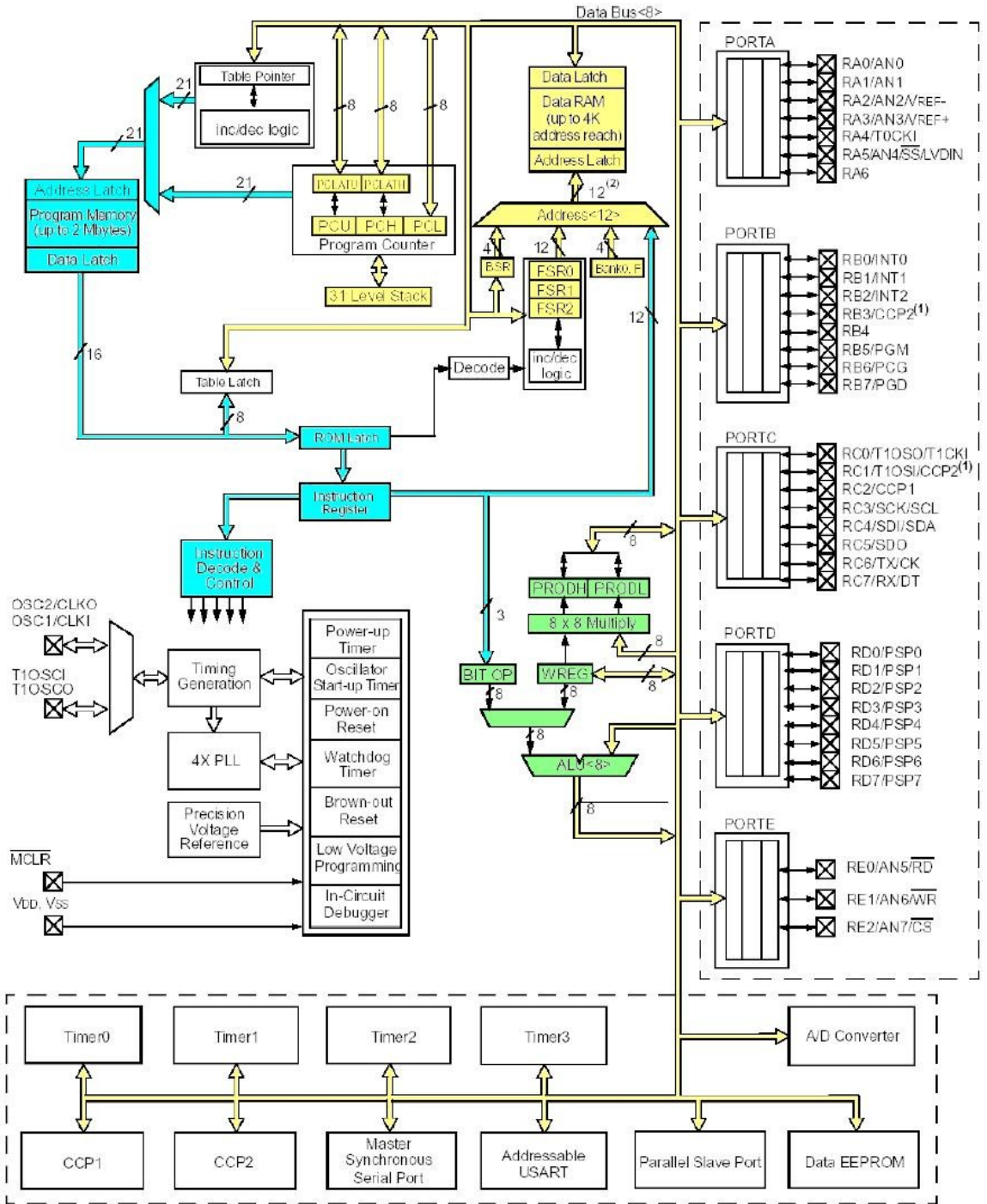
Harvard-arkkitehtuuri



- Erilliset käsky- ja dataväylät
- Käskymuisti: (Flash)
  - 16 bittinen väylä, 15 bittinen osoite (max 21 bittiä)
  - 32 kilotavua, 16k sanaa (1k = 1024)
- Datamuisti: (SRAM) lisäksi 256 tavua EEPROM
  - 8 bittinen väylä, 12 bittinen osoite
  - 1,5 kilotavua (maksimi 4 kilotavua)

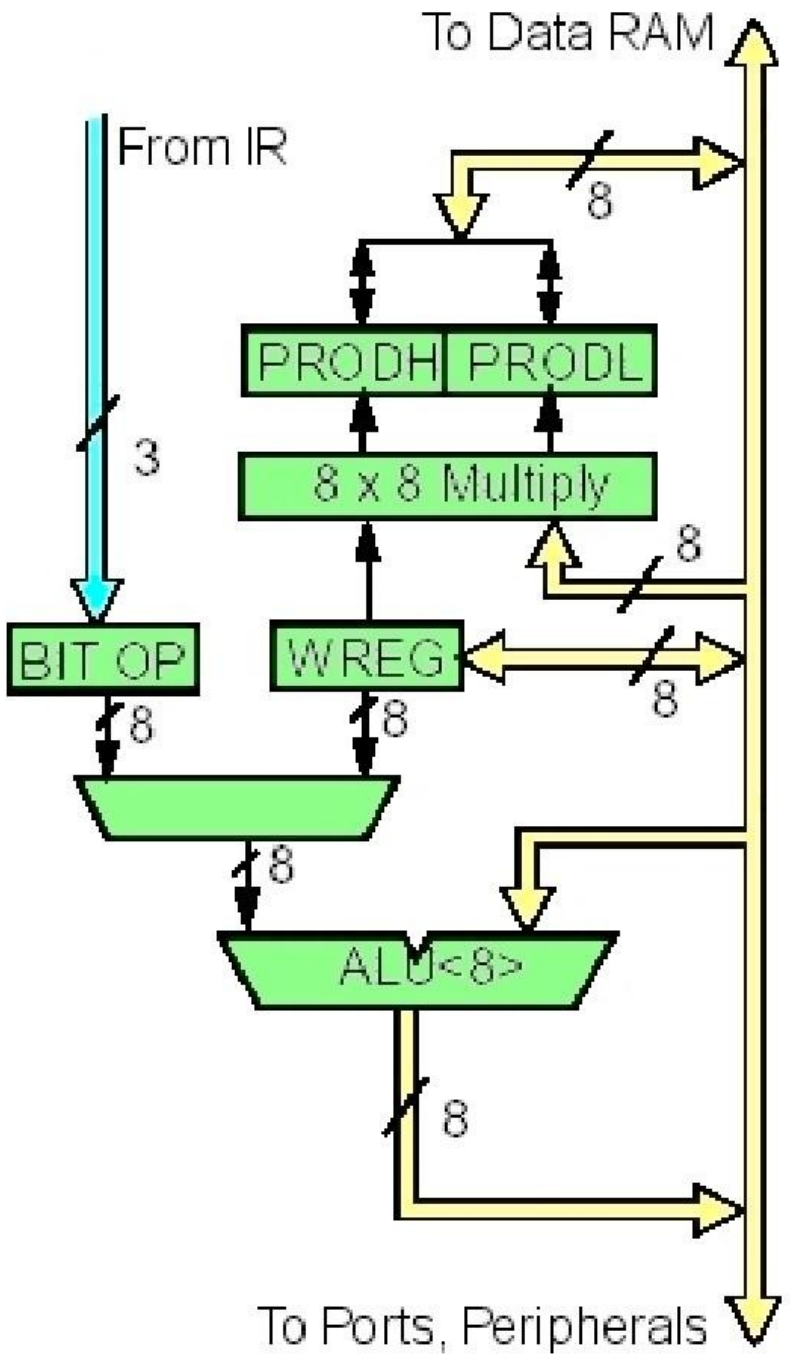
18F4x2 Block Diagram

- 16-bit wide instruction bus (blue)
- 8-bit wide data bus (yellow)

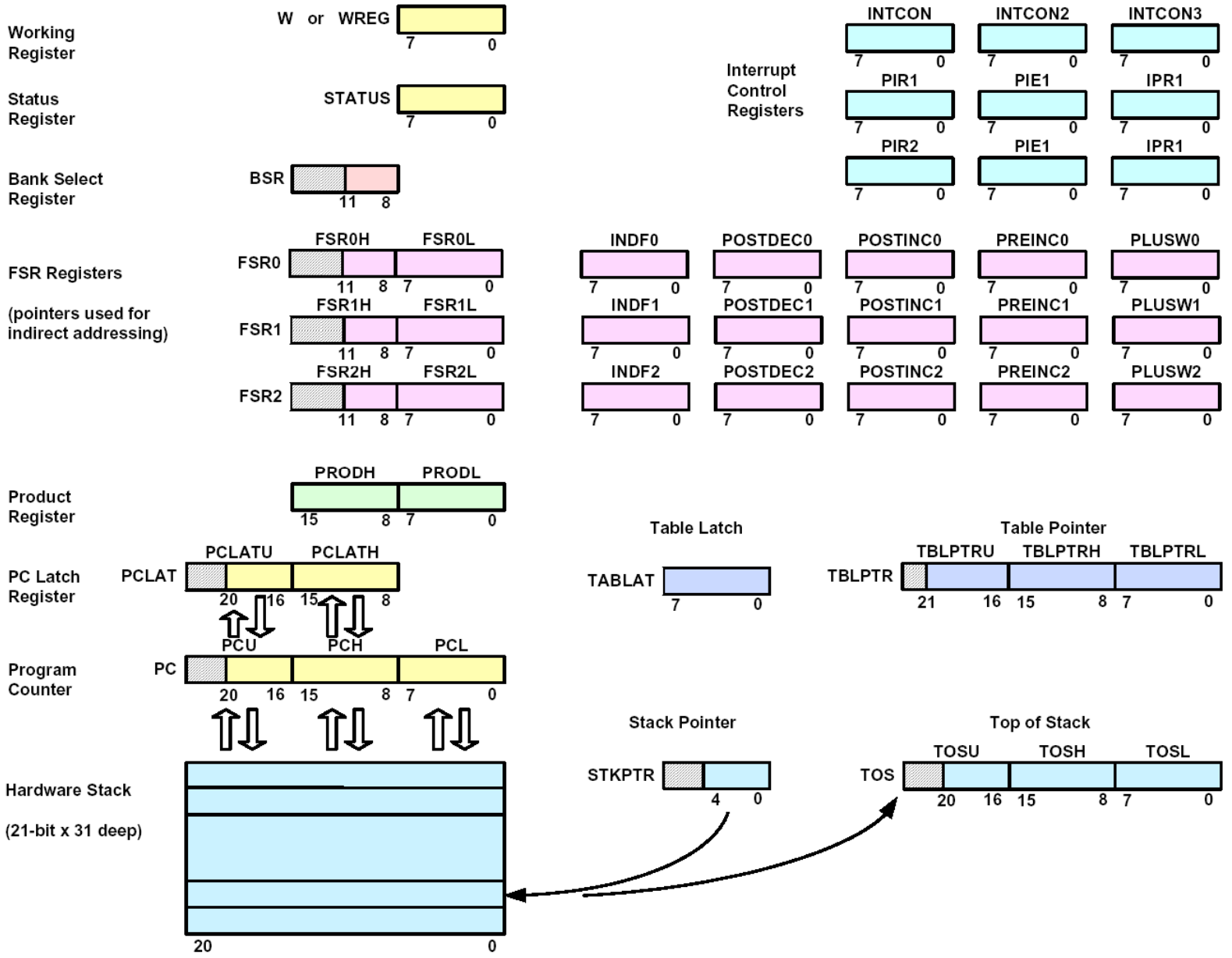


ALU

- 8 bittinen
- Työregisteri (WREG) toimii ”akkuna”
- Operandien haku koko Data-muistista, myös SFR:t
- Rekisteristä rekisteriin arkkitehtuuri
- 8x8 kertolasku
  - noin 70 kertaa nopeampi, kuin algoritminen kertolasku
- Matematiikkakirjasto luokan koneilla (Math18)



Tärkeimmät rekisterit (SFR:t)



STATUS rekisteri

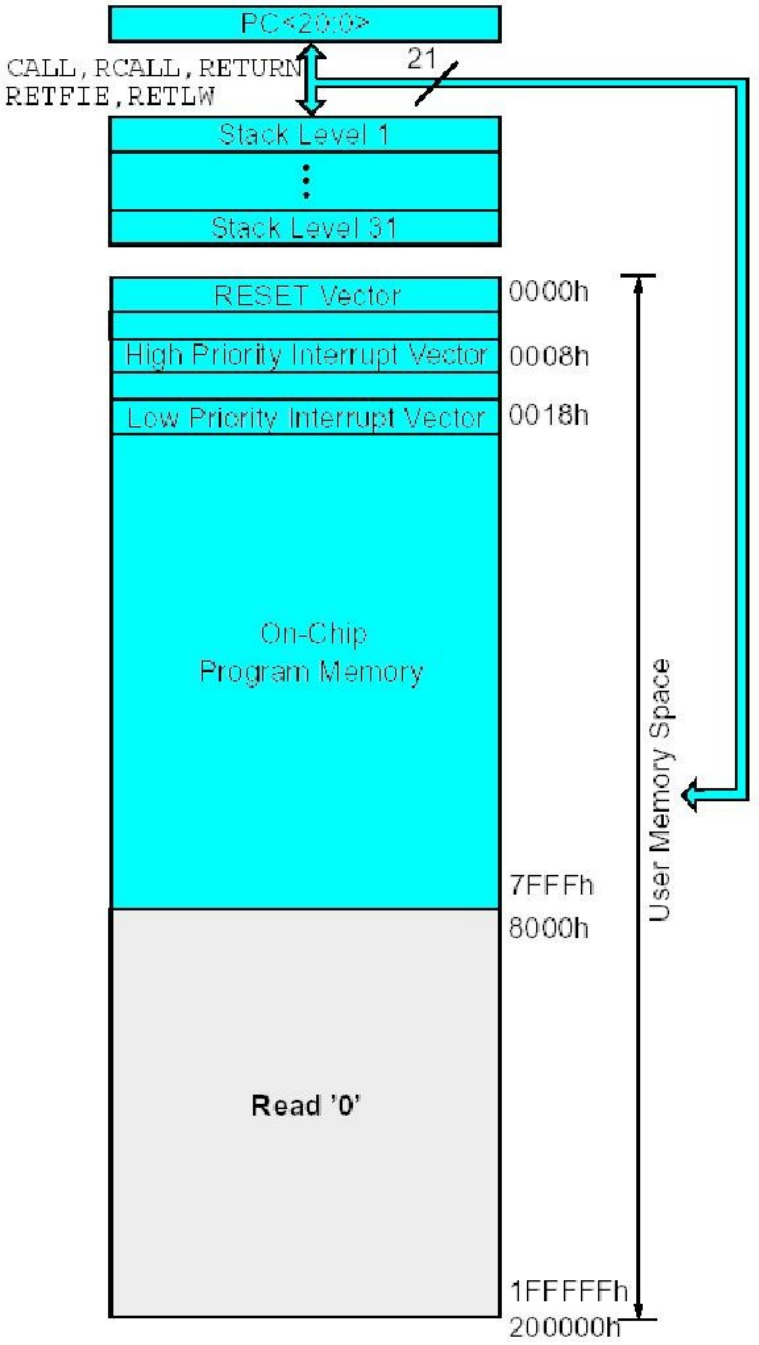
STATUS: Status (Flags) Register **0xFD8**

p. 52	U-0	U-0	U-0	R/W	R/W	R/W	R/W	R/W
Bit Names	-	-	-	N	OV	Z	DC	C
	bit 7							bit 0
PO/BO Reset	-	-	-	x	x	x	x	x
MCLR Reset	-	-	-	u	u	u	u	u

- Bitit 7-5: Ei implementoitu, palauttavat ’0’:n
- Bitti 4: ’1’ jos ALU:n tulos negatiivinen
- Bitti 3: ’1’ jos ALU:ssa tapahtui ylivuoto (signed)
- Bitti 2: ’1’ jos ALU:n tulos on nolla
- Bitti 1: ’1’ jos ”ylivuoto” 4. ja 5. bitin välissä (digit)
- Bitti 0: ’1’ jos ALU:ssa tapahtui ylivuoto (unsigned)
- Bittien 0,1 ja 3 polariteetti vaihtuu, jos kysessä vähennyslasku

Käskymuisti

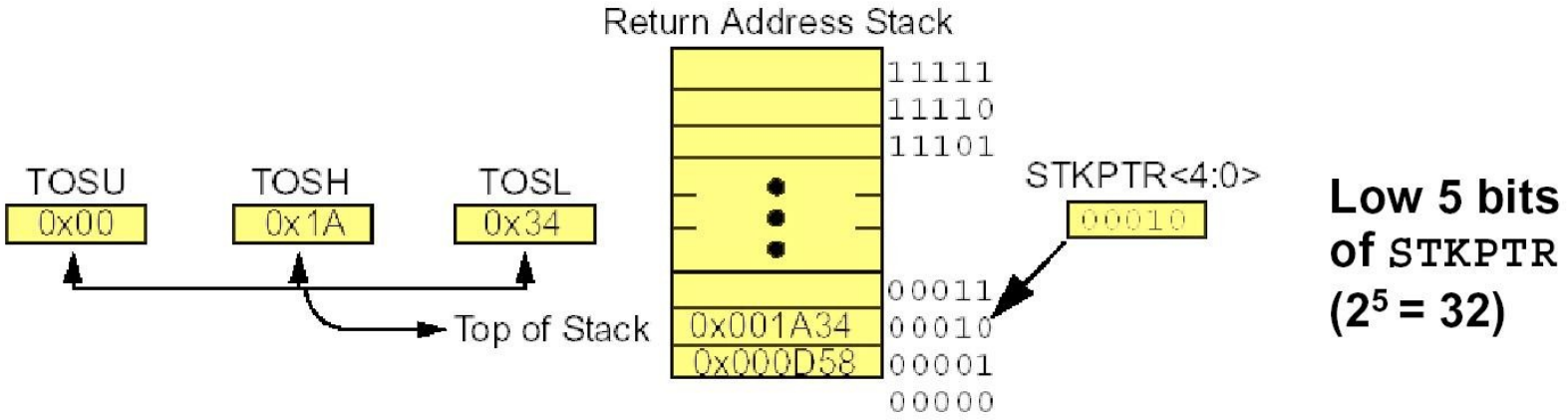
- 21 bittinen ohjelmalaskuri
  - Voi osoittaa 2 megatavua
- 18F452:ssa on 32 kilotavua käskymuistia (16k sanaa)
- Reset -osoite on 0x000000
- Keskeytysosoitteet:
  - korkea prioriteetti 0x000008
  - matala prioriteetti 0x000018
- Implementoimaton muisti palauttaa ’0’:n, eli NOP



Ohjelmalaskuri (PC)

- 21 bittiä kolmessa rekisterissä:
  - PCU : PCH : PCL
- Vain PCL suoraan luettavissa / kirjoitettavissa
- Muita käytetään PCLATU:n ja PCLATH:n kautta
- Kirjoitus PCL:ään kopioi myös PCLATU:n PCU:hun ja PCLATH:n PCH:hon
- Luku vastaavasti toiseen suuntaan
- CALL, RCALL, GOTO ja ehdolliset hypyt modifioivat PC:tä suoraan, ilman PCLATx rekistereitä

Paluuosoitepino



- Ohjelmalaskurin arvo talletetaan pinoon keskeytysten ja aliohjelmakutsujen yhteydessä
  - CALL ja RCALL
- Ohjelmalaskuriin ladataan talletettu arvo palattaessa
  - RETURN, RETLW ja RETFIE
- TOS rekistereissä STKPTR:n osoittama PC:n arvo
- 31 tasoa

STKPTR rekisteri

STKPTR: Return Stack Pointer Register **0xFFC**

p. 38	R/W	R/W	U-0	R/W	R/W	R/W	R/W	R/W
Bit Names	STKFUL	STKUNF	-	STKPTR4	STKPTR3	STKPTR2	STKPTR1	STKPTR0
	bit 7							bit 0
PO/BO Reset	0	0	-	0	0	0	0	0
MCLR Reset	0	0	-	0	0	0	0	0

- Bitti 7: ’1’ jos pinon täynnä
- Bitti 6: ’1’ jos alivuoto
- Bitti 5: Ei implementoitu, palauttaa ’0’:n
- Bitit 4-0: Pino-osoitin (0-31)
- Huom. bitit 7 & 6 täytyy nollata ohjelmallisesti tai tehdä POR (Power On Reset, alkunollaus)

Datamuisti

- Staattista RAM:ia (SRAM)
- 12 bittinen osoiteavaruus (max 4kilotavua)
- 18F452:ssa 1,5 kilotavua fyysistä muistia
  - Loput 2,5 kilotavua palauttavat ’0’:n
- Muisti jaettu ”pankkeihin”, jotka ovat kooltaan 256 tavua -> 8 bittiset osoitteet pankin sisällä
- Pankki valitaan asettamalla BSR<3:0>

Datamuistin sisältö

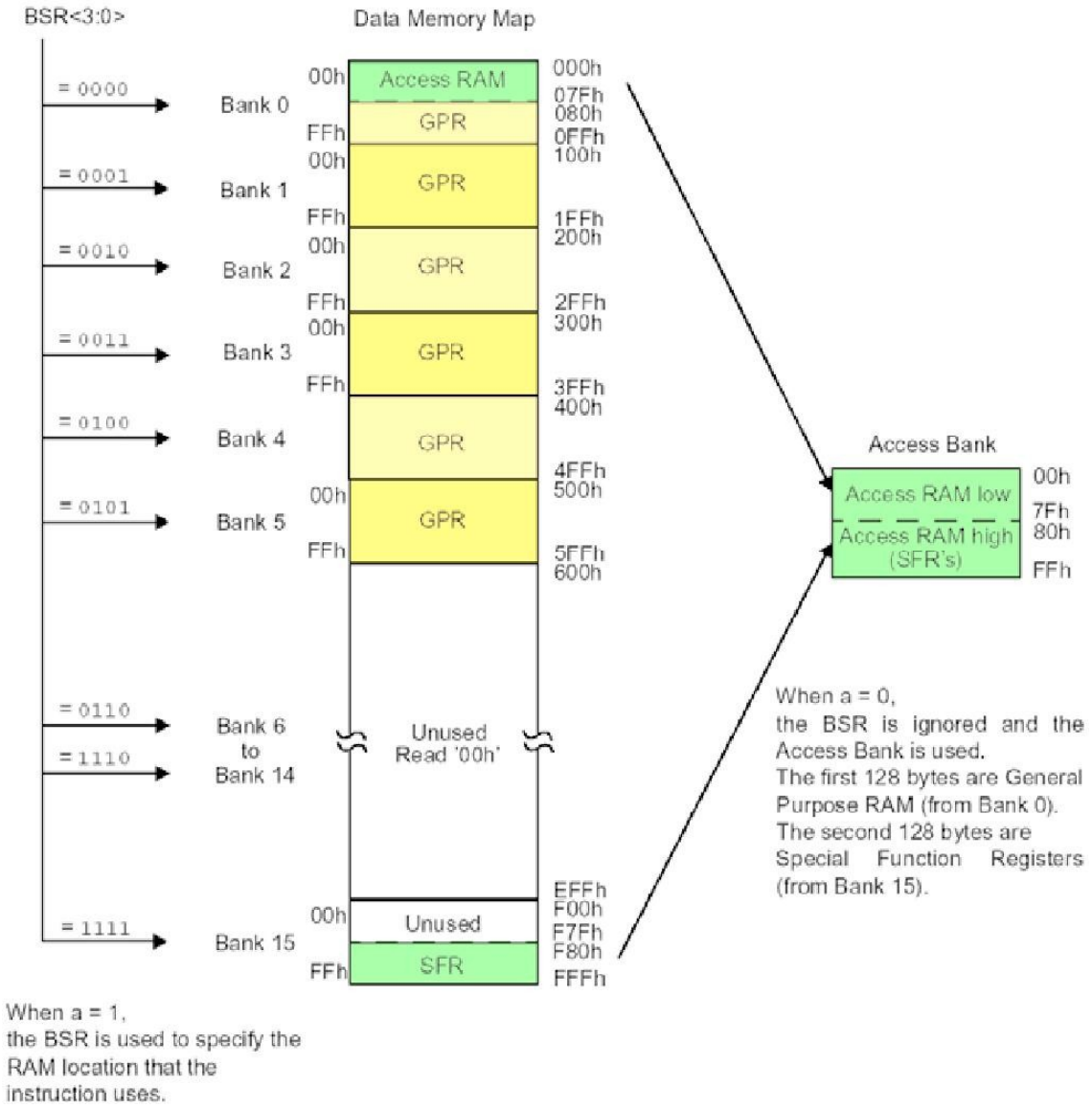
- Erikoisrekisterit (SFR:t)
  - Pankissa 15 ja access-pankissa
  - Osoitteet 0x000FFF – 0x000F80
  - Status, sisäänrakennettujen oheislaitteiden ohjaus
  - I/O portit ja niiden ohjaus
  - Lisää myöhemmin...
- Yleisrekisterit (normaali työmuisti), RAM
  - Kaikki implementoidut rekisterit käytettävissä ohjelmoijalle
  - Välitulosten ja muuttujien tallennukseen

Datamuistin osoittaminen

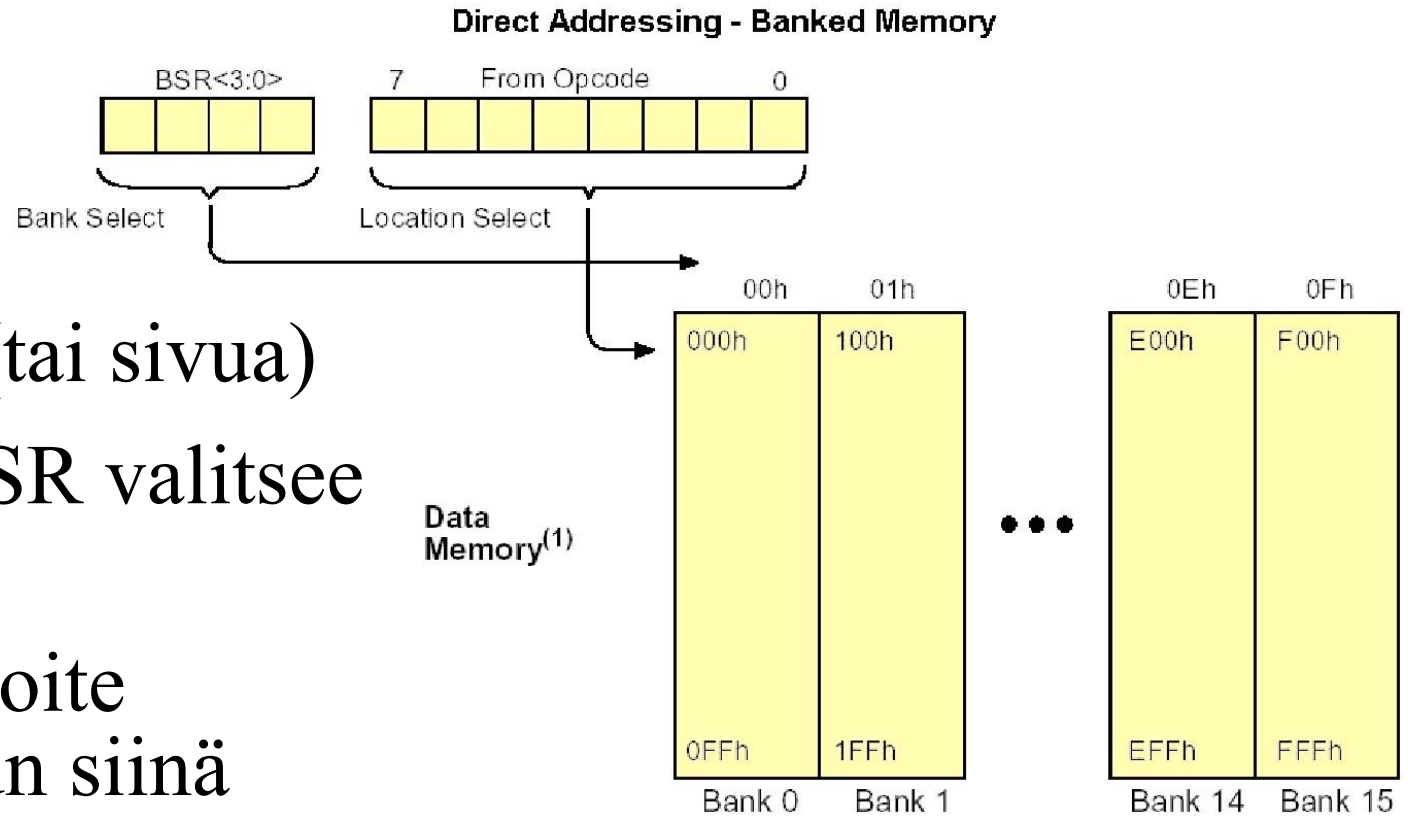
- Suora osoitus, osoite on kiinteä suorituksessa
  - 18F452:ssa 3 tapaa osoittaa muistia suoraan:
    - Pankitettu osoitus
    - Access-pankki
    - Osoite kokonaisuudessaan komennossa
- Epäsuora osoitus, osoite lasketaan suorituksen aikana
  - Käytetään osoitinta (=osoiterekisteriä)
  - 18F452:ssa yksi tapa epäsuoraan osoitukseen
    - Monia optioita

Muistikartta

- Suora osoitus (pankitettu)
  - 4 ylintä bittiä BSR rekisterissä
  - loput 8 käskyn mukana
  - => 12 bittiä
- Suora osoitus (access pankki)
  - alimmat 128 tavua pankista 0 (GPR)
  - ylimmät 128 tavua pankista 15 (SFR)
  - valitaan ’a’ bitillä
  - osoitteet aina 8 bittisiä

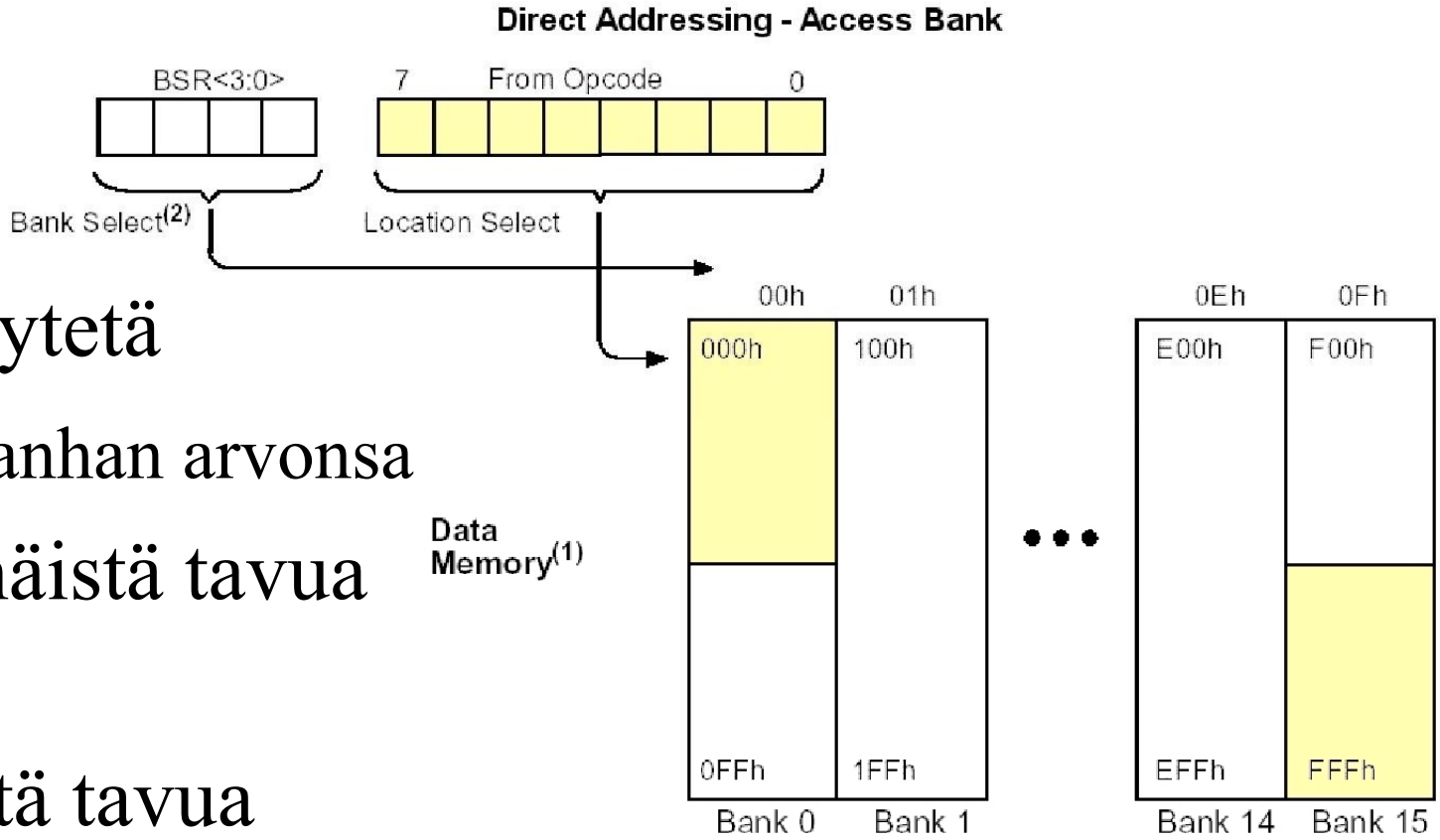


Suora osoitus (pankitettu)



- 16 pankkia (tai sivua)
- 4 bittinen BSR valitsee pankin
- 8 bittinen osoite valitsee tavun siinä pankissa (256:sta)
- 4+8=12 bittiä => 4 kilotavun osoiteavaruus

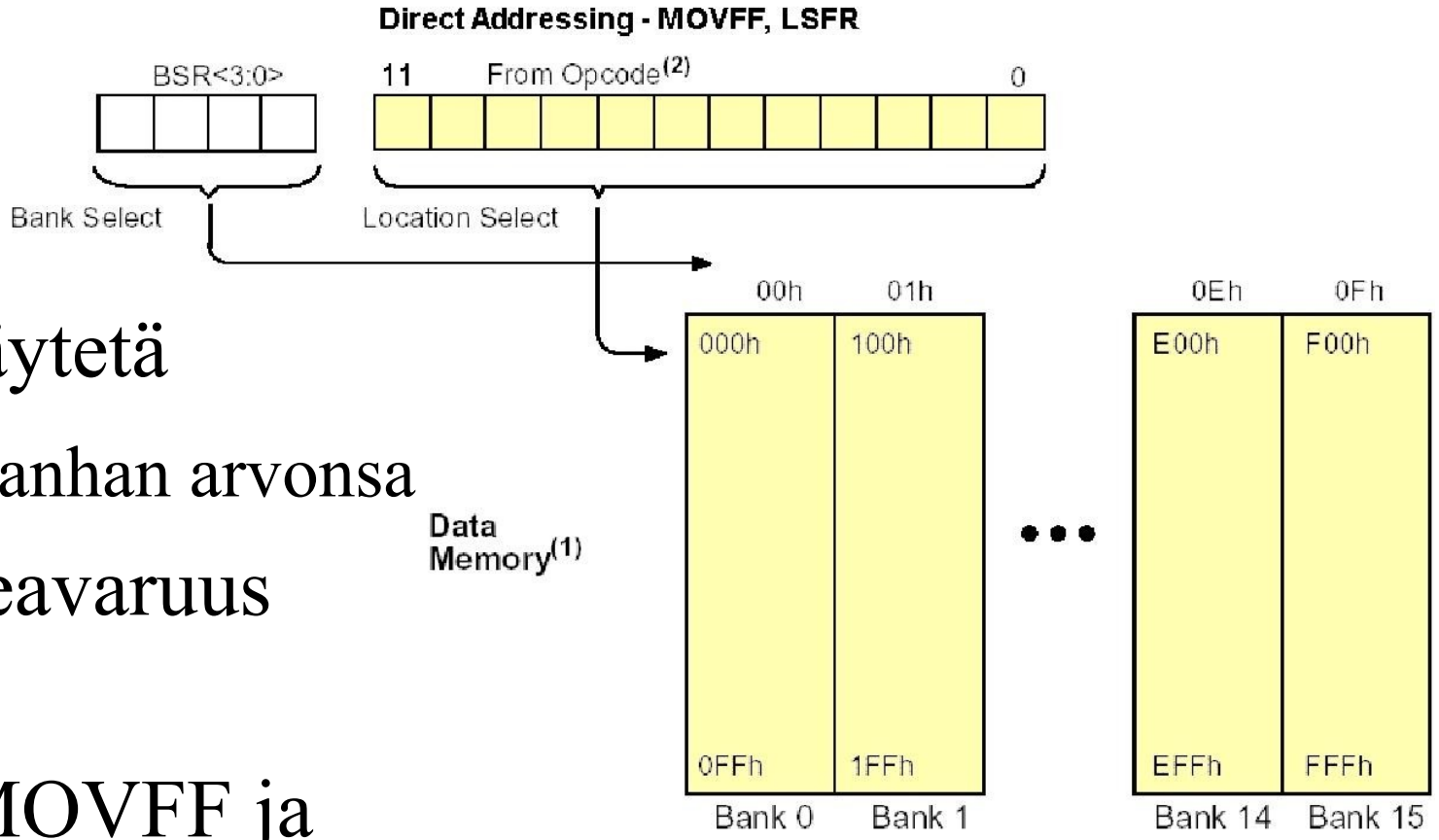
Suora osoitus (access pankki)



- BSR:ä ei käytetä
  - säilyttää vanhan arvonsa
- 128 ensimmäistä tavua pankista 0
- 128 viimeistä tavua pankista 15
  - erikoisrekisterit (SFR)

Suora osoitus (osoite komennossa)

- BSR:ä ei käytetä
  - säilyttää vanhan arvonsa
- Koko osoiteavaruus käytössä
- 2 käskyä: MOVFF ja LFSR



Suora osoitus (esimerkki)

- Useat käskyt ovat muotoa:
  - MNEMONIC, f, d, a
    - f = 8 bittinen osoite rekistereihin (0x00 – 0xFF)
    - d = kohde (d=’0’ => tulos WREG:iin, d=’1’ => tulos takaisin f-rekisteriin)
    - a = access pankki (a=’0’ => access, a=’1’ => pankitettu)

```
MOVLB   02              ; set BSR to Bank 2
ADDWF   H’55’, 0, 1     ; ADD WREG to contents of addr. 55 (f=55)
                        ; in bank 2 (a=1), result to WREG (d=0)
```

- Operandin osoite on siis 0x0255. Jos taas

```
                        ; MOVLB not required
ADDWF   0xAA, 0, 0      ; ADD WREG to contents of addr. AA (f=AA)
                        ; in access bank (a=0), result to WREG (d=0)
```

- Operandin osoite on 0x0FAA (0xAA>0x7F => pankki 15)

Epäsuora osoitus

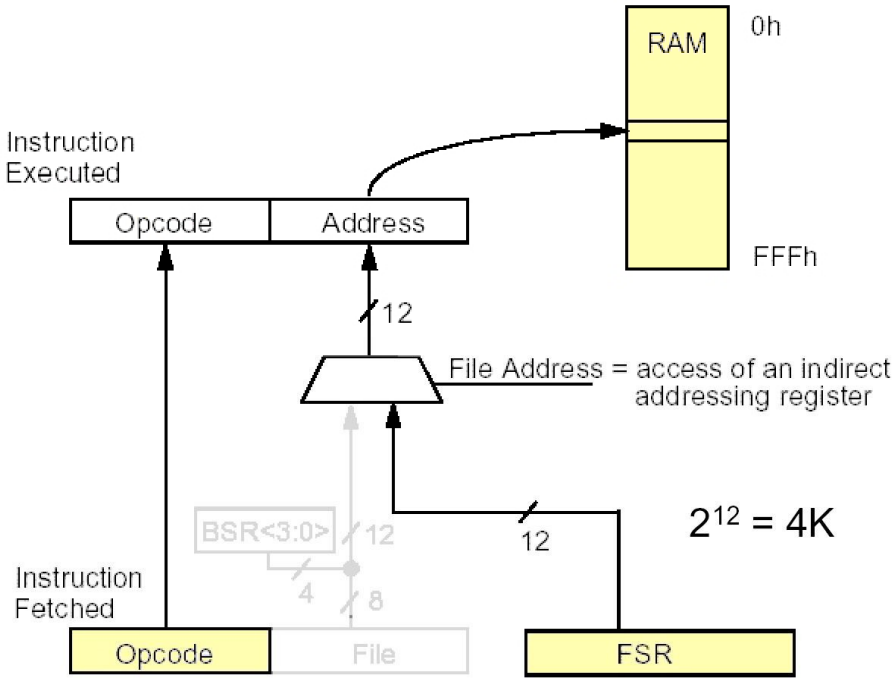
- Osoite lasketaan suorituksen aikana
- Osoittamiseen käytetään osoittimia (FSR)
  - 18F452:ssa 3 FSR:ä (kaksi osaisia, 12 bittiä)
- Itse data saatavilla FSR:ä vastaavassa INDF:ssä
- FSR0H=0x00 ja FSR0L=0x0A osoite kokonaisuudessaan osoite on 0x000A, ja tuon osoitteen sisältö on käytettävissä INDF0 rekisterissä

```
LFSR    FSR0, ADDR   ; load FSR0 with value ADDR
MOVWF   INDF0, 0     ; move w to location pointed by FSR0
```

- INDF0 on epäsuora osoitus
- Siihen viittaaminen operoi sitä muistipaikkaa, johon FSR0 viittaa

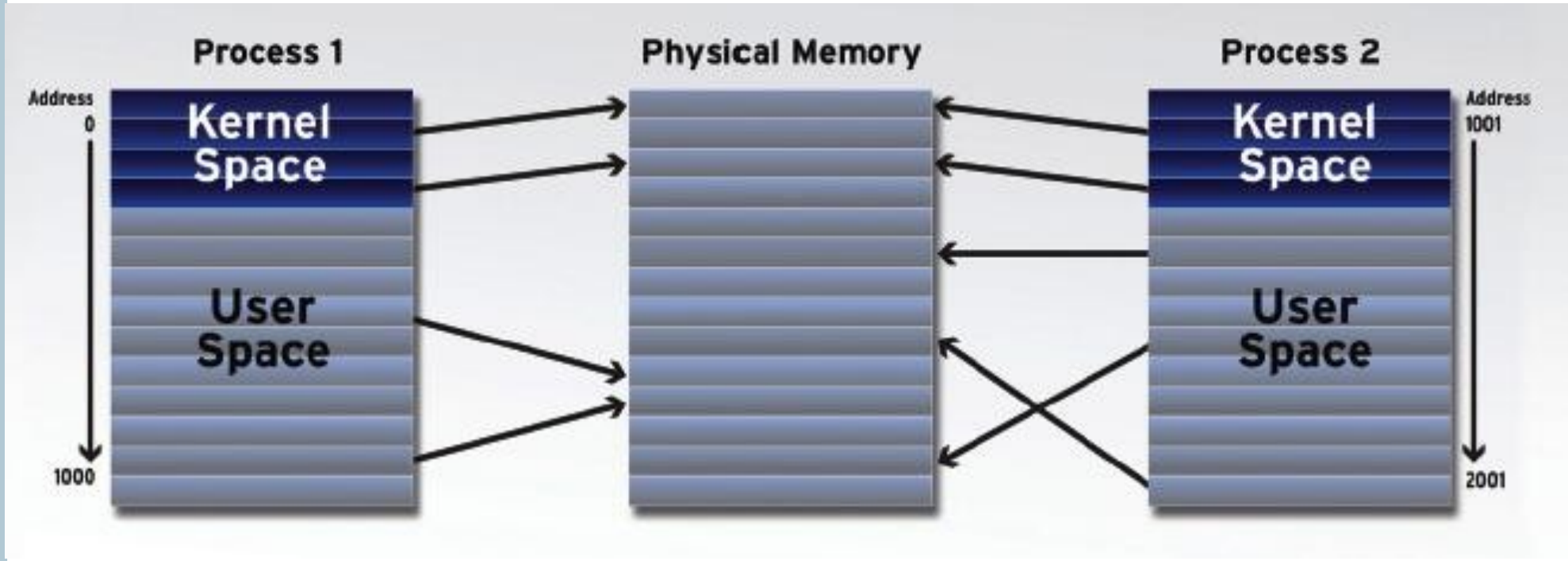
Epäsuora osoitus

- Pointteriin voidaan kohdistaa esi-lisäys, jälki-lisäys tai jälki-vähennys
  - Askellus taulukoissa
- Voidaan myös asettaa off-set, WREG:in arvo, ei muuta FSR:n arvoa




```
MOVWF   INDF0, W        ; don't change the pointer value FSR0
MOVWF   POSTDEC1, W     ; post-decrement FSR1
MOVWF   POSTINC2, W     ; post-increment FSR2
MOVWF   PREINC0, W      ; pre-increment FSR0
MOVWF   PLUSW1, W       ; offset value in FSR1 by value in WREG
```

Virtuaaliosoitteista (data)



- Kaikki ohjelmat luulevat, että niillä on käytössä koko muisti
  - Poislukien käyttöjärjestelmä

Virtuaaliosoitteista (data)

- PIC-sarjan prosessoreissa ei ole MMU:ta eli muistinhallintayksikköä joka voisi tarjota ohjelmille virtuaaliosoitteita
  - Normaalissa tietokoneessa ajettavat ohjelmat voidaan ladata rinnakkain koska ohjelmat saavat omat virtuaaliosoiteavaruudet joita prosessorin MMU hallinnoi
- MMU puuttuu myös monista muista pienistä prosessoreista / mikrokontrollereista
- Käyttöjärjestelmät, esim. Linux, vaativat MMU:n
  - Voidaan kuitenkin tehdä oma “käyttöjärjestelmä”

Virtuaaliosoitteista (data)

- Omassa käyttöjärjestelmässä MMU:n tehtävät hoidetaan jakamalla osoiteavaruus etukäteen ohjelmien kesken
  - PIC:issä annetaan ohjelmille oma pankki muistista
    - Tai jos ohjelma vaatii useampia pankkeja, sekin voidaan järjestää (tosin monimutkaisuus kasvaa merkittävästi)
  - Käyttöjärjestelmän virkaa hoitava koodi vaihtaa pankin oikeaksi, ennenkuin se jatkaa suoritusta tietyssä ohjelmassa
  - Kaikki ohjelmat täytyy kirjoittaa niin, etteivät ne muuta BSR:n arvoa
    - Toki myös muita rajoitteita, esimerkiksi tiettyä ajastinta voi käyttää vain yksi ohjelma kerrallaan, mutta nyt puututaan vain datan osoittamiseen…
  - Usein myös epäsuoran osoituksen rekistereistä yksi varataan “käyttöjärjestelmälle”, yksi keskeytyksille ja yksi jää ohjelmien käyttöön

Virtuaaliosoitteista (data)

- Kuitenkaan muistin suojausta eri ohjelmien välillä ei voida toteuttaa mitenkään
  - Normaalissa tietokoneessa ohjelmat eivät pääse lukemaan, saatikka kirjoittamaan, toistensa muistialueita
  - Myöskin “käyttöjärjestelmän” muistialue on avoinna kaikille ohjelmille

Heittovaihtotiedosto (engl. Swap)

- Koska pienissä prosessoreissa ei ole MMU:ta, on swapin toteuttaminen todella hankalaa
  - Voidaan kuitenkin toteuttaa samaan tyyliin kuin virtuaaliosoitteet
  - Kontrollointi muuttuu todella haastavaksi, eikä tällaista yleensä harrasteta
  - Mieluummin käytetään suoraan ulkoista muistilaitetta niin, että ladataan vain tarvittu data jonnekin (työalueeksi varattuun muistipankkiin) ja kun sitä on käsitelty, siirretään tulokset takaisin massamuistiin

Seuraavaksi:

- Käskykanta tarkemmin
- Assembler
  - Ohjemoinnin perusteet
  - Työkalujen käyttö