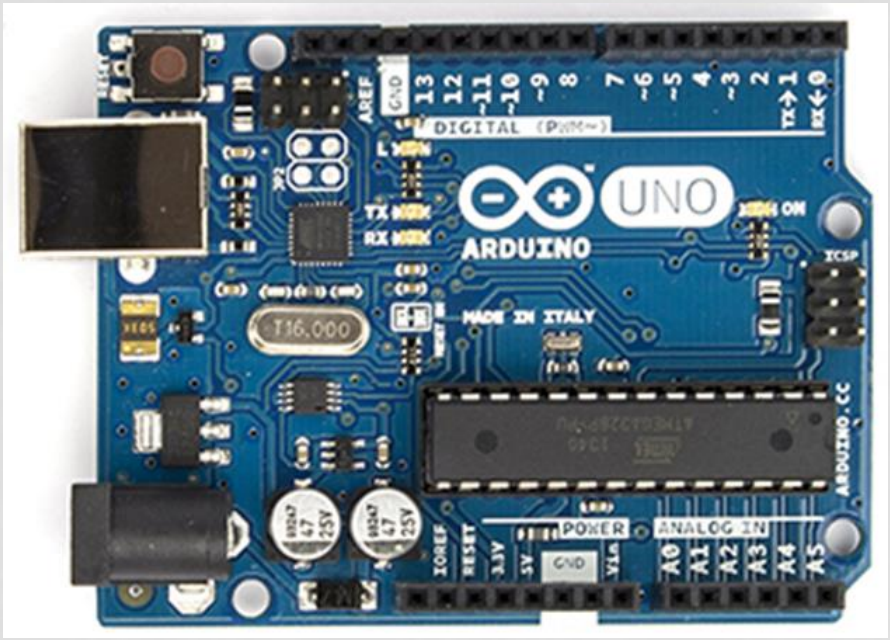


CS

WEEK 2-2

AGENDA

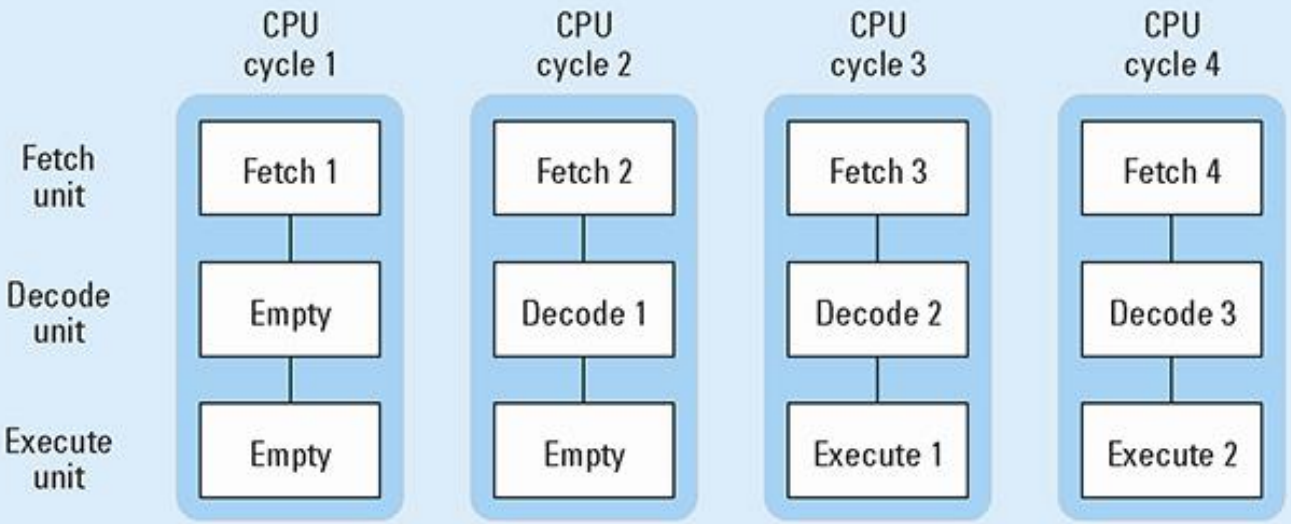
week	onderwerp	P&H	AT	Dijkstra
1	coderingen en talstelsels representatie van getallen optellen en aftrekken vermenigvuldigen en delen logische poorten schakelingen met poorten geheugen-elementen systeemklok & timers	App. B2, B3, B7, B8, B9 2.4 3.2 t/m 3.5 app B	App. A, B 3.1, 3.2, 3.3	H1 H2
2	typen computers 8 great ideas organisatie van de computer CPU intern, instructies uitvoeren geheugen systeem adres- en databus byte ordering pipelining de AVR MCU	1.1 t/m 1.4 2.12 4.1 t/m 4.5	1.3 2.1, 2.2 3.7	H3 6.1 en 6.2 7.1 en 7.2
3	typen geheugen caching opslag (ssd, harddisk) translating and starting a program parallelle architecturen - h/w multi-threading - multicore - GPU	5.2, 5.3 6.4 t/m 6.6	2.2, 2.3 7.3, 7.4 H8	4.1 7.3

AGENDA

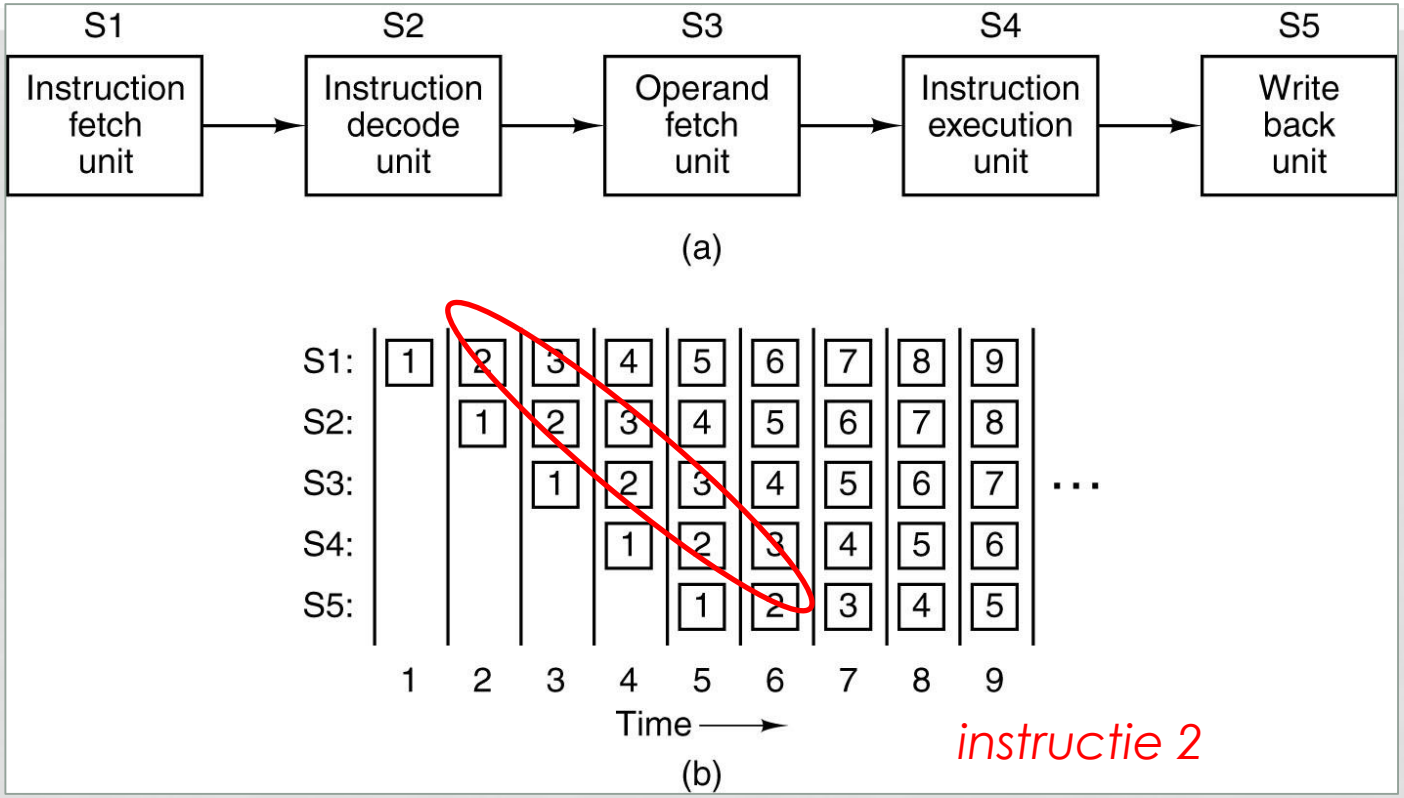
- **pipelining**
- adres- en databus
- Harvard en Von Neumann
- byte ordering
- de AVR MCU

PIPELINING

Figuur 7.6 **Pipelining**



PIPELINING



- a five-stage pipeline
- the state of each stage as a function of time; nine clock cycles are illustrated

PROBLEMEN BIJ PIPELINING

- interlocking
  - afhankelijkheid tussen instructies in de pipeline
  - instructie n heeft resultaat van instructie n-1 nodig
- load en store bezetten databus te lang
- conditional branching
  - heel vervelend, volgende instructie is moeilijk te voorspellen
  - branch prediction
- un-conditional branching
  - ook lastig, want pas na decoderen is bekend welke instructie het is, en dan is de fetch (van de volgende) al geladen
  - NOP's invoegen

RISC

- assembly vs. C
  - eenvoudige of complexe instructies ?
- RISC : Reduced Instruction Set Computer
  - eenvoudige laag-niveau instructies
  - dataoverdracht geheugen alleen via LOAD en STORE
  - alle andere operaties werken alleen op interne registers
  - veel interne registers
  - (bijna) alle instructies duren één clockcycle
  - zodat pipelining optimaal kan worden toegepast, want fasen cyclus zijn even lang

AGENDA

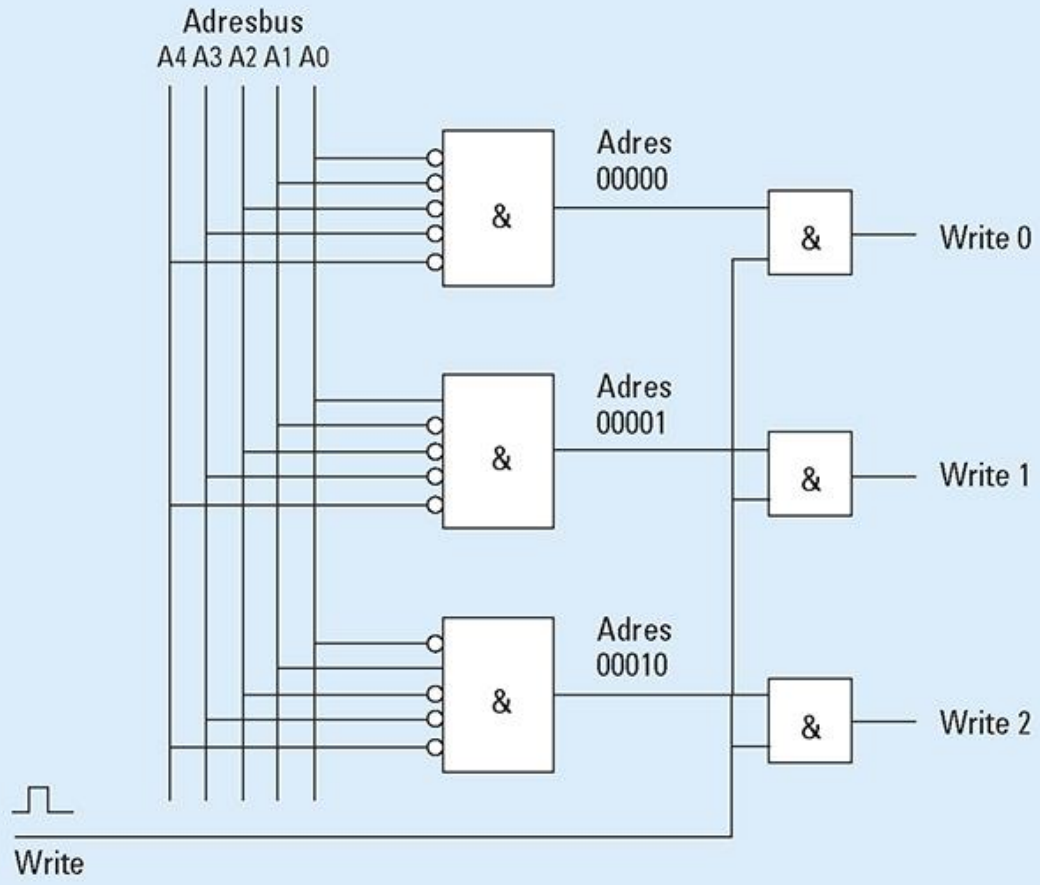
- pipelining
- **adres- en databus**
- Harvard en Von Neumann
- byte ordering
- de AVR MCU

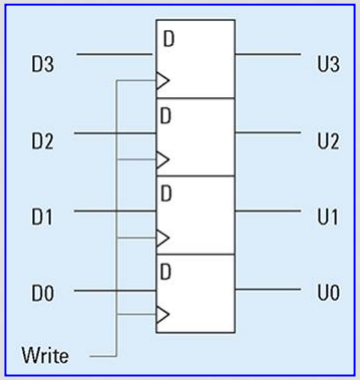
ADRESSELECTIE

- geheugen van computer bestaat uit grote verzameling registers
- hoe kunnen we één register selecteren ?
  - elk register heeft eigen adres
  - afhankelijk van waarde op adresbus wordt één register geselecteerd (geadresseerd)

ADRESSELECTIE

Figuur 2.28 **Selectie**





ADRESSELECTIE

Onthouden :

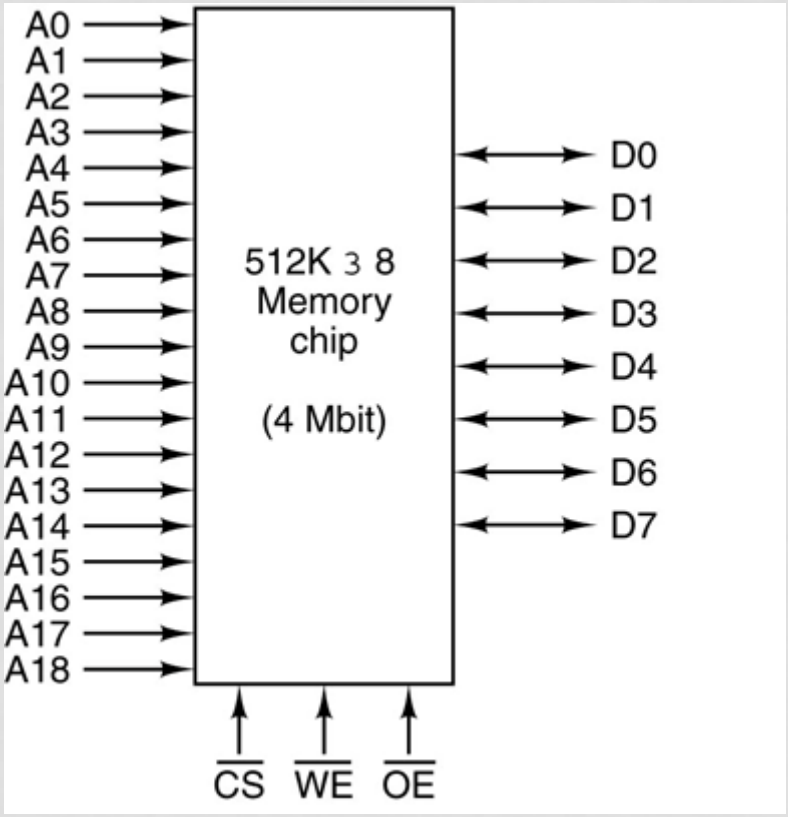
- k = kilo = 1024 = 2^{10}
- M = mega = 1024*1024 = 2^{20}
- G = giga = 1024*1024*1024 = 2^{30}

ADRESSELECTIE

- stel de adresbus bestaat uit 4 lijnen, hoe groot is het geheugen (# adressen) dat ik kan bereiken ?
- zelfde vraag voor een adresbus van 11 lijnen

- stel een geheugenkaart heeft 64 adressen, hoeveel adreslijnen zijn nodig ?
- zelfde vraag voor een geheugenkaart van 256M adressen

GEHEUGEN CHIP



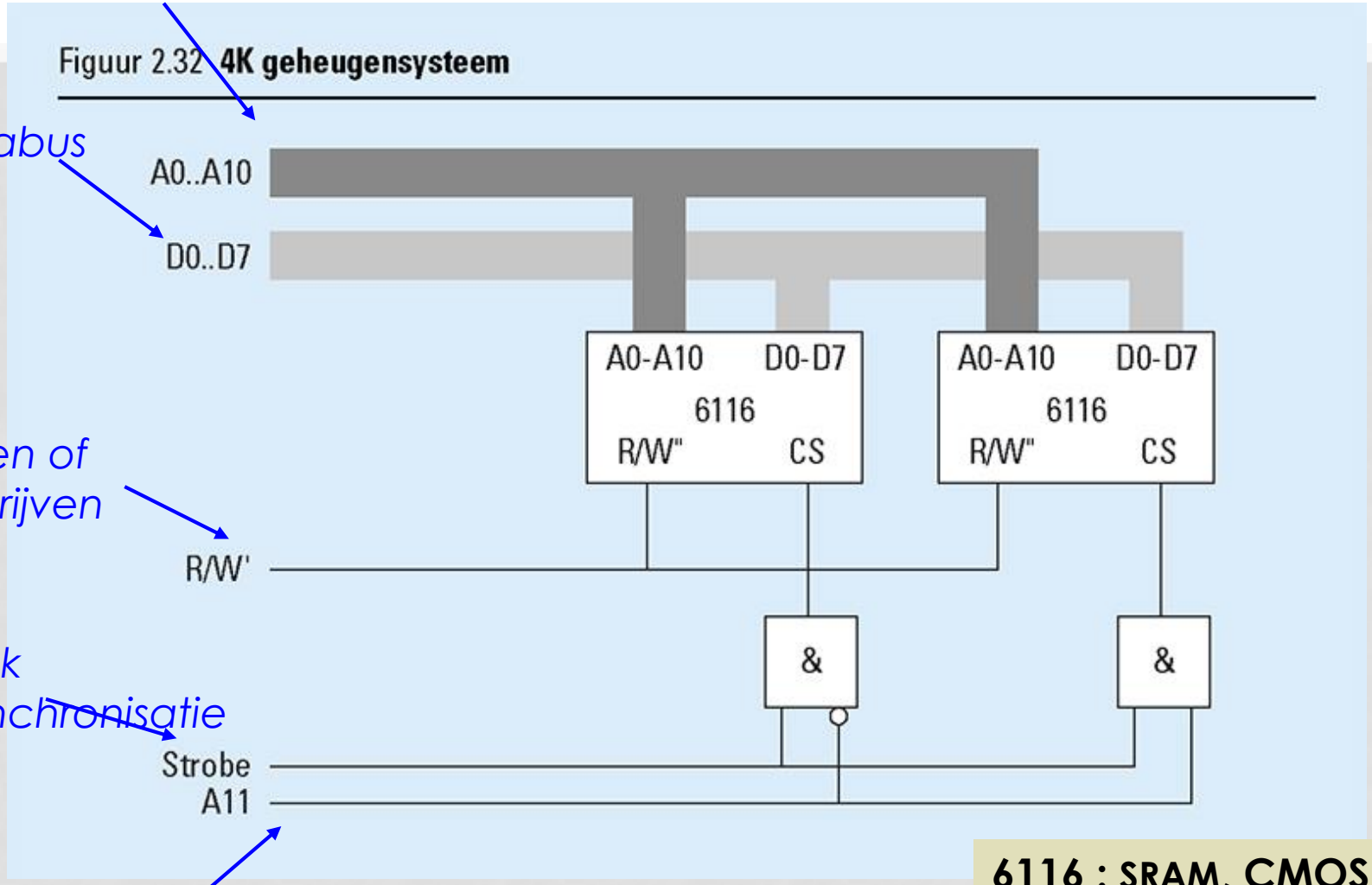
control lines :

CS = Chip Select (kan als extra adreslijn worden gebruikt)
WE = Write Enable
OE = Output Enable

GEHEUGENSYSTEEM

adresbus

databus



lezen of schrijven

klok synchronisatie

= CS (welke helft van het geheugen)

6116 : SRAM, CMOS, 2k

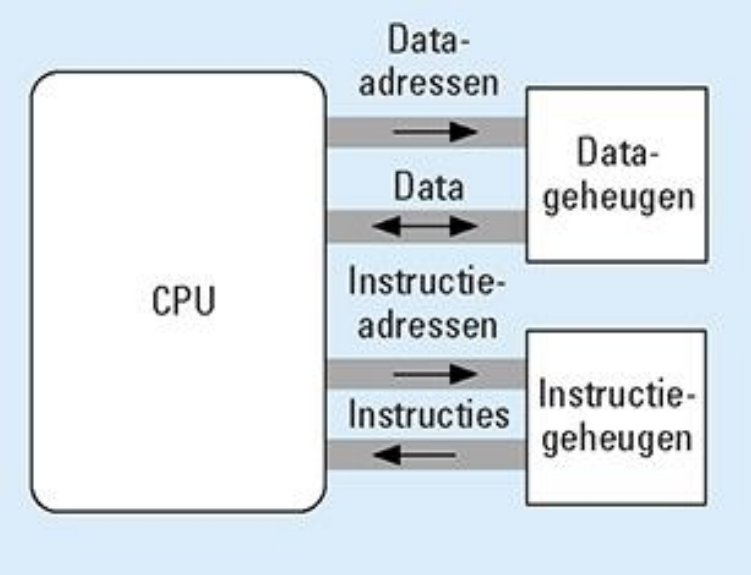
AGENDA

- pipelining
- adres- en databus
- **Harvard en Von Neumann**
- byte ordering
- de AVR MCU

VON NEUMANN FLESSENHALS

- Von Neumann architectuur :
  - één geheugen voor programma en data
  - alles moet over zelfde databus

HARVARD ARCHITECTUUR



- programma heeft eigen geheugen, en data heeft eigen geheugen
- elk met eigen databus
- noodzakelijk bij pipelining (meerdere instructies tegelijk verwerken)

AGENDA

- pipelining
- adres- en databus
- Harvard en Von Neumann
- **byte ordering**
- de AVR MCU

BYTE ORDERING

adres 0x100	0x101	0x102	0x103
0x80	0x00	0x00	0x00

- 4 byte integer
- is dit een groot of een klein getal ?

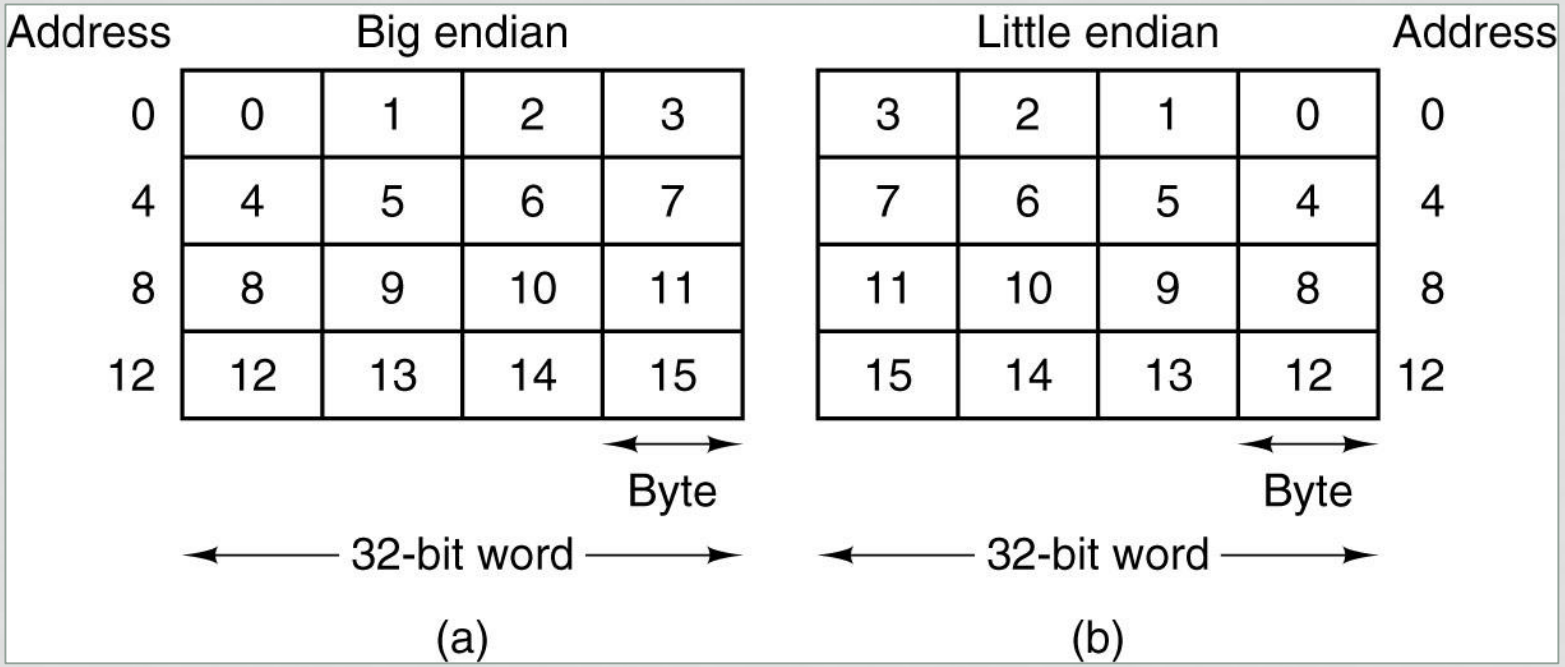
BIG END FIRST

adres 0x100	0x101	0x102	0x103
0x80	0x00	0x00	0x00

- big endian : hier staat een groot getal (unsigned)
- big endian : most significant byte op lage adres
- netwerken gebruiken big-endian order, dit heet daarom ook network order

wat is het probleem ?

BYTE ORDERING



(a) Big endian memory
(b) Little endian memory

BYTE ORDERING

Little Endian	Big Endian	Bi-Endian
Intel x86 and x86-64 series ARM before version 3 6502 (including 65802, 65C816) Z80 (including Z180, eZ80 etc.) MCS-48, 8051 DEC Alpha Altera Nios II Atmel AVR SuperH VAX, PDP-11	Motorola 6800 and 68k series SPARC before 9 Xilinx Microblaze IBM POWER IBM System/360, System/370 IBMESA/390, z/Architecture. The PDP-10	ARM 3 and above PowerPC Alpha SPARC V9 MIPS PA-RISC SuperH SH-4 IA-64

Bi-endian : capability of the machine to compute or pass data in either endian format. Many of these architectures can be switched via software to default to a specific endian format.

AGENDA

- pipelining
- adres- en databus
- Harvard en Von Neumann
- byte ordering
- **de AVR MCU**

CPU EN MCU

- MCU = Micro Controller Unit
- = CPU + diverse typen geheugen + diverse bussen + seriële I/O op één enkele chip
  - ook wel "computer op een chip"
  - typisch gebruikt in embedded applicaties

POPULAIRE 8 BIT MCU'S

- PIC (Microchip)
- 8051 (Intel)
- AVR (Atmel)

- wat is een 8 bit MCU ?
  - registers in de CPU zijn 8 bit breed
- intellectual property (IP) licensing
  - ARM, MIPS, enz. hebben intellectueel eigendom maar maken zelf geen chips
  - ze verkopen licenties aan halfgeleider fabrikanten en ondersteunen hun product ontwikkeling met kennis en tools

AVR

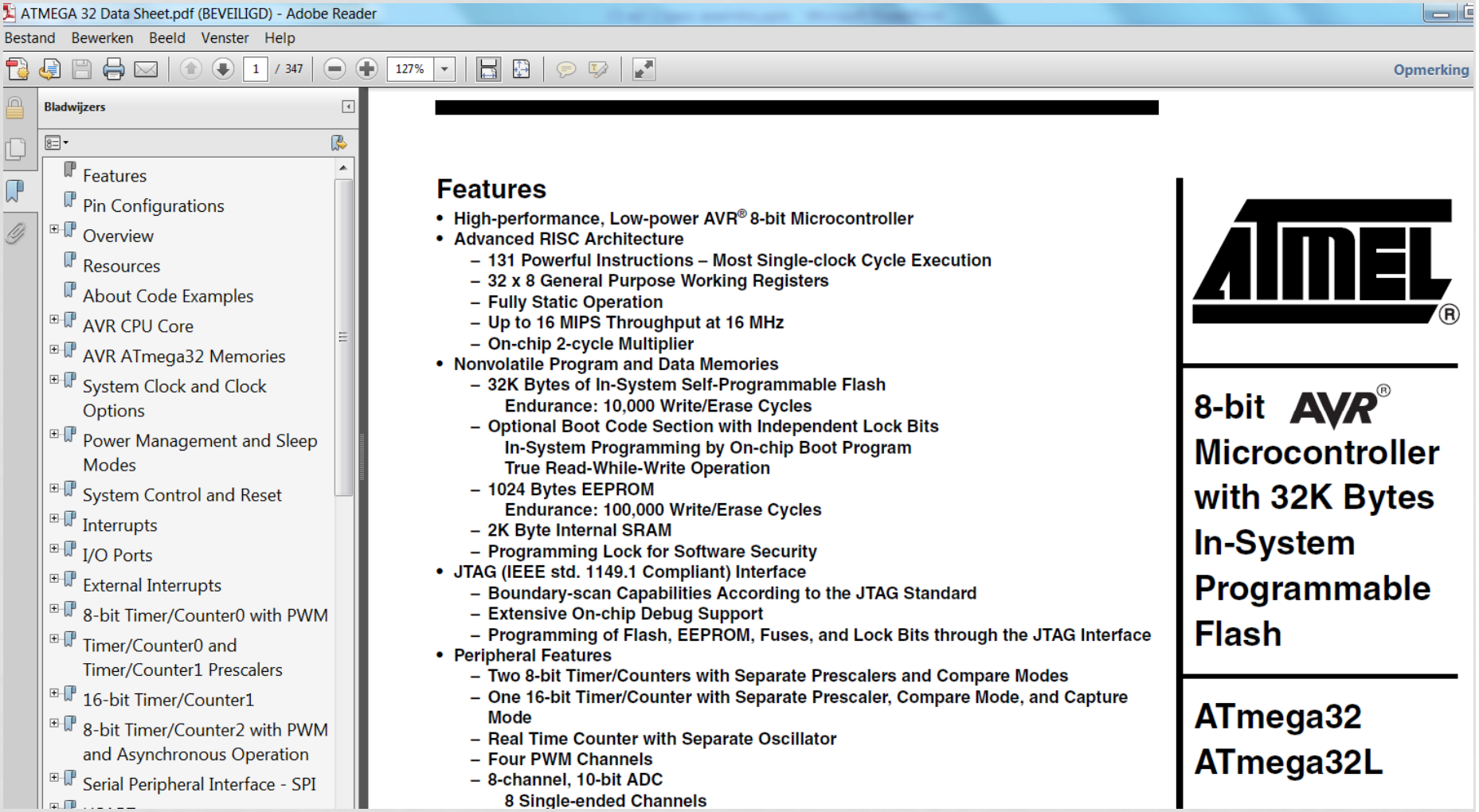
- the AVR architecture was conceived by two students at the Norwegian Institute of Technology
  - Alf-Egil Bogen & Vegard Wollan
  - AVR : **A**lf **V**egard **R**isc
- modified Harvard architecture
- 8-bit RISC single chip microcontroller
- developed by Atmel in 1996 (headquarters San Jose, California)
- one of the first microcontroller families to use on-chip flash memory for program storage

AVR

- Atmel AVR 8-bit MCU's zijn er in diverse configuraties en behuizingen
  - tinyAVR, MegaAVR, XMAGA, USB AVR, Automotive AVR, ...
  - zelfde instructie set en register set
- wij gebruiken ATmega32

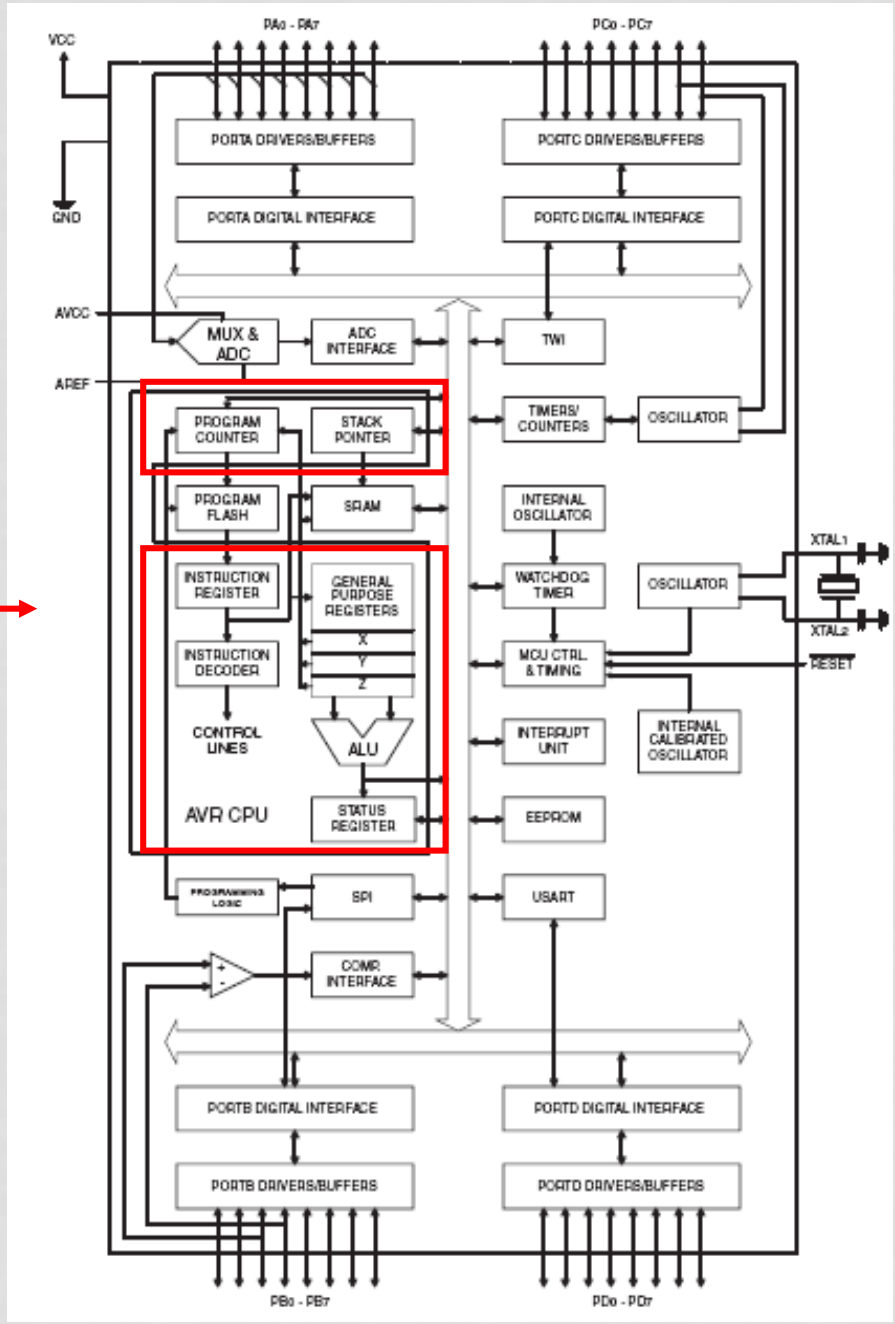
ATMEGA32 SPECS

- 131 instructies
- 32 8-bit GP registers
- up to 16 MIPS throughput
- 32K programmable flash (code)
- 2K interne SRAM
- 1K EEPROM
- timer/counters, seriële & parallelle I/O, ADC


ATMEGA 32 Data Sheet.pdf (BEVEILIGD) - Adobe Reader
Bestand Bewerken Beeld Venster Help
1 / 347
127%
Opmerking
Bladwijzers
Features
Pin Configurations
Overview
Resources
About Code Examples
AVR CPU Core
AVR ATmega32 Memories
System Clock and Clock Options
Power Management and Sleep Modes
System Control and Reset
Interrupts
I/O Ports
External Interrupts
8-bit Timer/Counter0 with PWM
Timer/Counter0 and Timer/Counter1 Prescalers
16-bit Timer/Counter1
8-bit Timer/Counter2 with PWM and Asynchronous Operation
Serial Peripheral Interface - SPI
Features
• High-performance, Low-power AVR® 8-bit Microcontroller
• Advanced RISC Architecture
– 131 Powerful Instructions – Most Single-clock Cycle Execution
– 32 x 8 General Purpose Working Registers
– Fully Static Operation
– Up to 16 MIPS Throughput at 16 MHz
– On-chip 2-cycle Multiplier
• Nonvolatile Program and Data Memories
– 32K Bytes of In-System Self-Programmable Flash
Endurance: 10,000 Write/Erase Cycles
– Optional Boot Code Section with Independent Lock Bits
In-System Programming by On-chip Boot Program
True Read-While-Write Operation
– 1024 Bytes EEPROM
Endurance: 100,000 Write/Erase Cycles
– 2K Byte Internal SRAM
– Programming Lock for Software Security
• JTAG (IEEE std. 1149.1 Compliant) Interface
– Boundary-scan Capabilities According to the JTAG Standard
– Extensive On-chip Debug Support
– Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
• Peripheral Features
– Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
– One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
– Real Time Counter with Separate Oscillator
– Four PWM Channels
– 8-channel, 10-bit ADC
8 Single-ended Channels
ATMEL
8-bit AVR® Microcontroller with 32K Bytes In-System Programmable Flash
ATmega32
ATmega32L

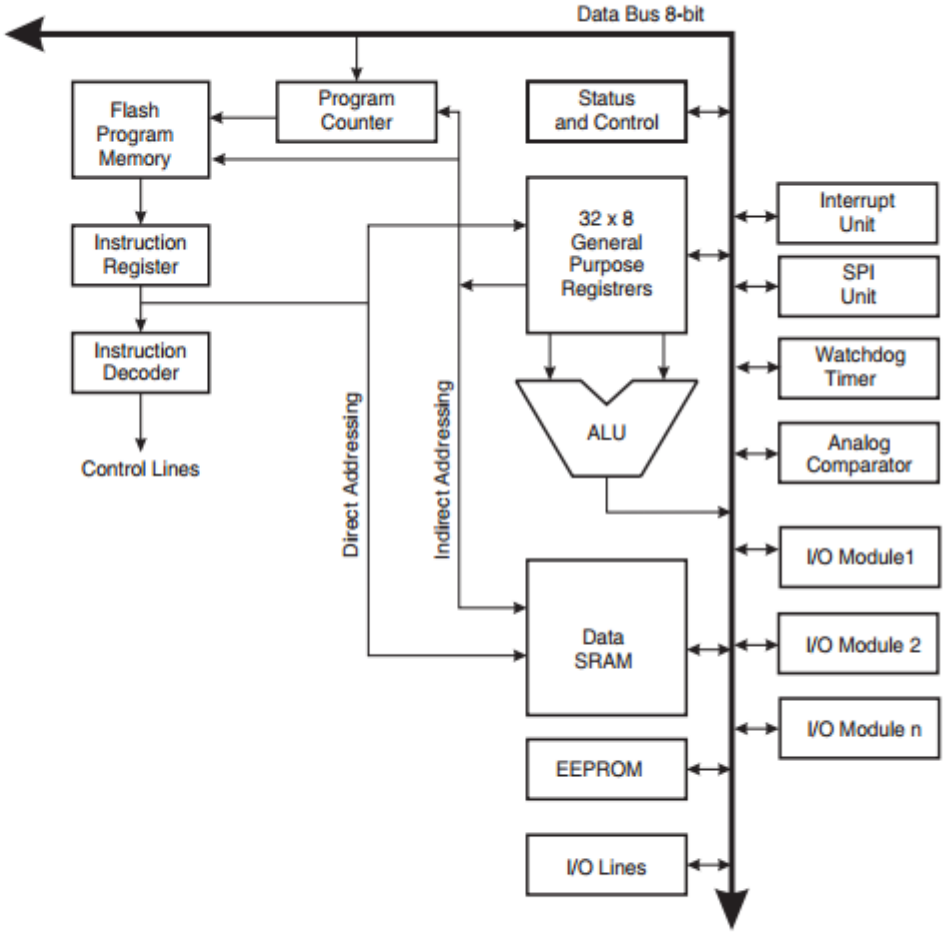
Atmega32 blokschema

CPU



Architectural Overview

Figure 3. Block Diagram of the AVR MCU Architecture



In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program