Datorsystem VT 2022

Stockholm University

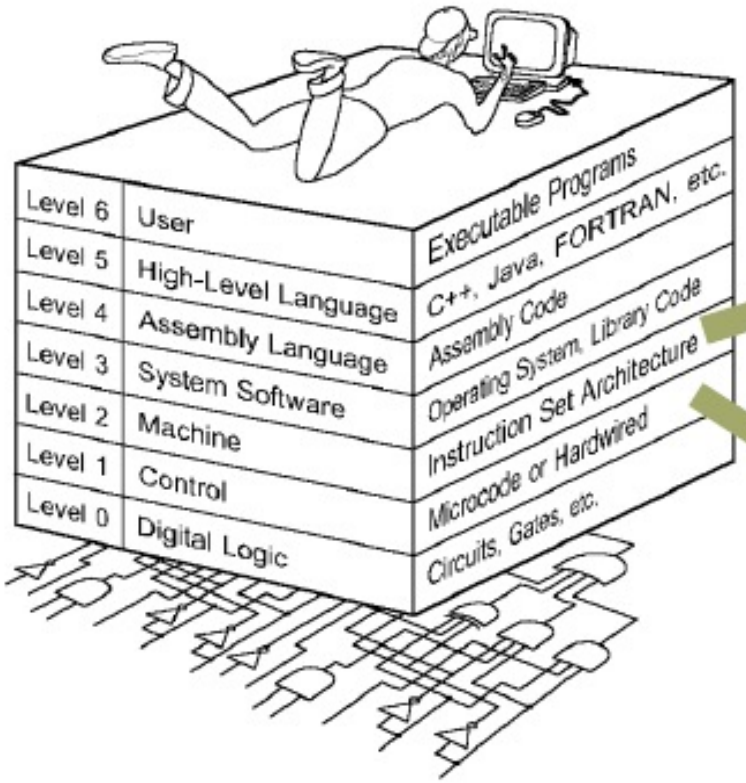
F4
Introduction to processor & Machine Architecture

Outline

Stockholm University

- Boolean Algebra
- Digital Logic
- CPU Basic
- The Bus
- CPU: executes program instructions
- Processor operation: NIOS II
- Interrupts
- Maskable & NonMaskable Interrupts

The Computer Level Hierarchy



Level 2: Machine Level

- Also known as the Instruction Set Architecture (ISA) Level.
- Consists of instructions that are particular to the architecture of the machine.
- Programs written in machine language need no compilers, interpreters, or assemblers.

Level 1: Control Level

- A *control unit* decodes and executes instructions and moves data through the system.
- Control units can be *microprogrammed* or *hardwired*.
- A microprogram is a program written in a low-level language that is implemented by the hardware.
- Hardwired control units consist of hardware that directly executes machine instructions.

Boolean Algebra

Stockholm University

- Boolean algebra is a mathematical system for the manipulation of variables that can have one of two values.
  - In **formal logic**, these values are "**true**" and "**false**."
  - In digital systems, these values are "on" and "off," 1 and 0, or "high" and "low"
- Boolean expressions are created by performing operations on Boolean variables.
  - Common Boolean **operators** include **AND**, **OR**, and **NOT**.

Boolean Algebra

- A ***Boolean operator*** can be completely described using a **truth table**.
- The truth table for the Boolean operators AND and OR are shown at the right.
- The **AND operator** is also known as a **Boolean product**. The **OR** operator is the **Boolean sum**.

X AND Y

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X OR Y

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Algebra

Stockholm University

- The truth table for the Boolean NOT operator is shown at the right.
- The **NOT** operation is most often designated by an overbar. It is sometimes indicated by a prime mark (‘) or an “elbow” (¬).

NOT x

x	$\overline{x}$
0	1
1	0

Boolean Algebra

Stockholm University

- A **Boolean function** has:
  - At least *one Boolean variable*,
  - At least *one Boolean operator*, and
  - At least one *input from the set* {0,1}.
- It produces an ***output*** that is also a **member of the set** {0,1}.

Now you know why the binary numbering system is so handy in digital systems.

Boolean Algebra

Stockholm University

- The **truth table** for the Boolean function:

  $F(x,y,z) = x\bar{z}+y$

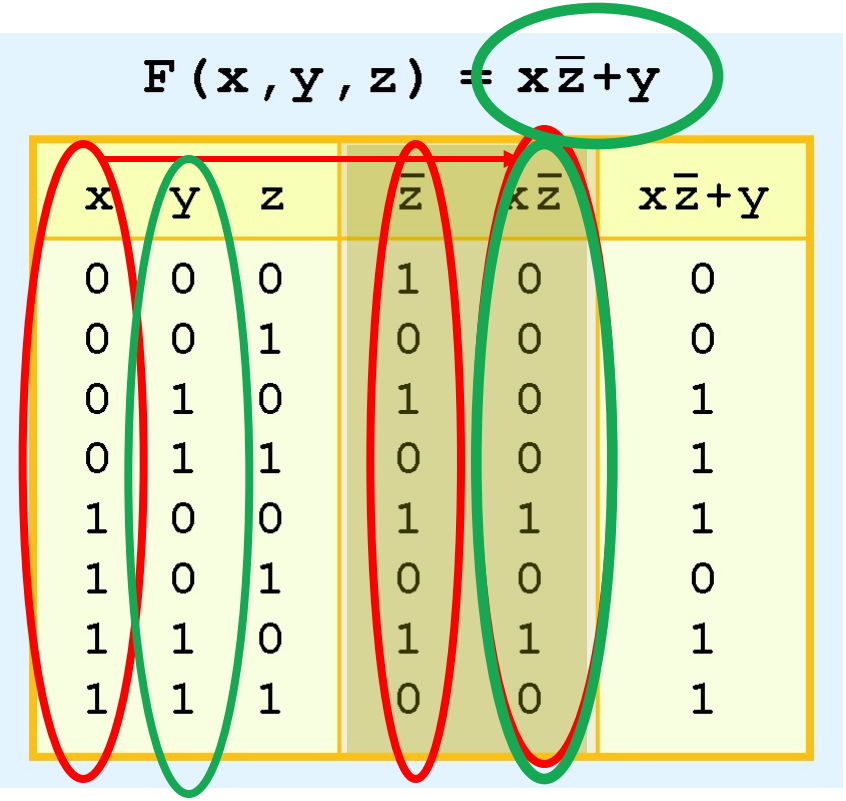
  is shown at the right.
- To make evaluation of the Boolean function easier, the truth table contains extra (shaded) columns to hold evaluations of subparts of the function.

$F(x,y,z) = x\bar{z}+y$

x	y	z	$\bar{z}$	$x\bar{z}$	$x\bar{z}+y$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

Boolean Algebra

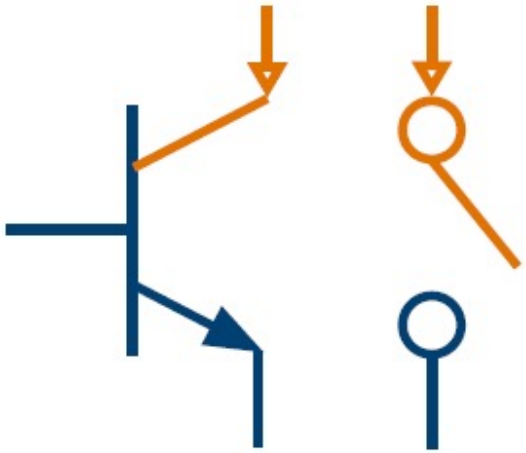
- As with common arithmetic, Boolean operations have rules of precedence.
- The NOT operator has highest priority, followed by AND and then OR.
- This is how we chose the (shaded) function subparts in our table.

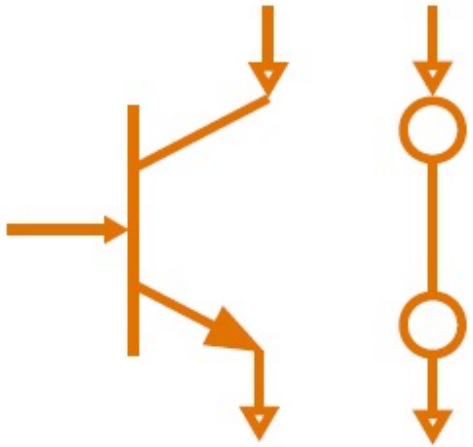


x	y	z	$\bar{z}$	$x\bar{z}$	$x\bar{z}+y$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

Stockholm University

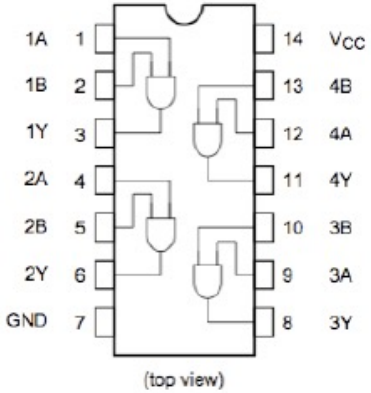
Transistorns funktion

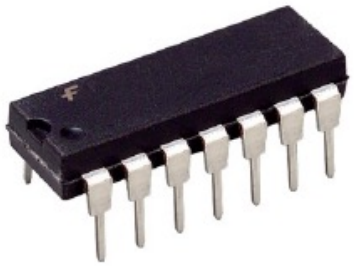




Digital logik med grindar

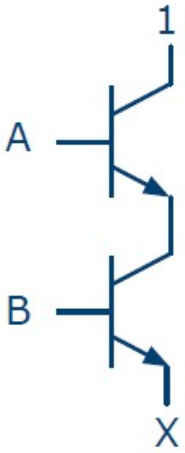
- Flera transistorer kan kombineras till grindar.
- Varje grind har en viss logisk funktion



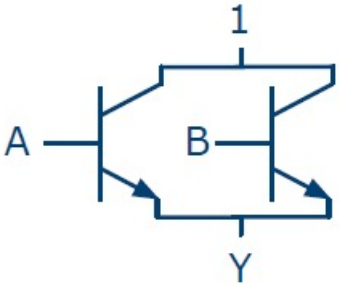


Stockholm University

Digital logik



X = 1 om A = 1 och B = 1
X = A och B
X = A * B

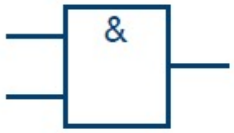


Y = 1 om A = 1 eller B = 1
Y = A eller B
Y = A + B

AND-grinden

A	B	Ut
0	0	0
0	1	0
1	0	0
1	1	1

IEEE

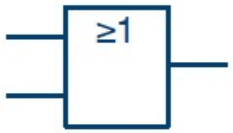


IEC

OR-grinden

A	B	Ut
0	0	0
0	1	1
1	0	1
1	1	1

IEEE

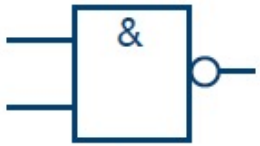


IEC

NAND-grinden

A	B	Ut
0	0	1
0	1	1
1	0	1
1	1	0

IEEE

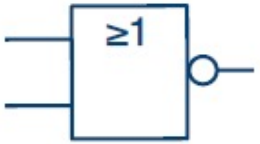


IEC

NOR-grinden

A	B	Ut
0	0	1
0	1	0
1	0	0
1	1	0

IEEE

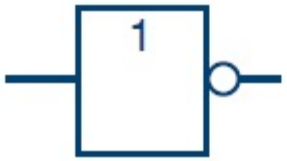


IEC

NOT-grinden

A	Ut
0	1
1	0

IEEE

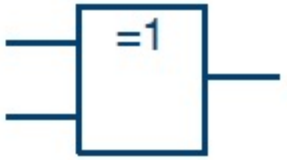


IEC

XOR-grinden

A	B	Ut
0	0	0
0	1	1
1	0	1
1	1	0

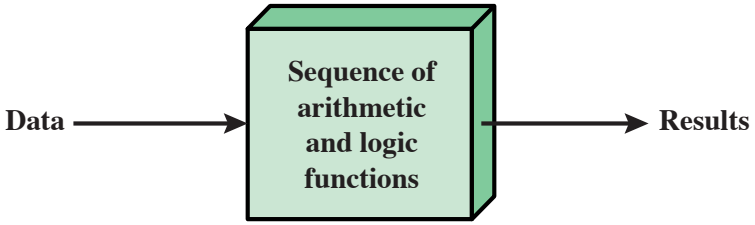
IEEE



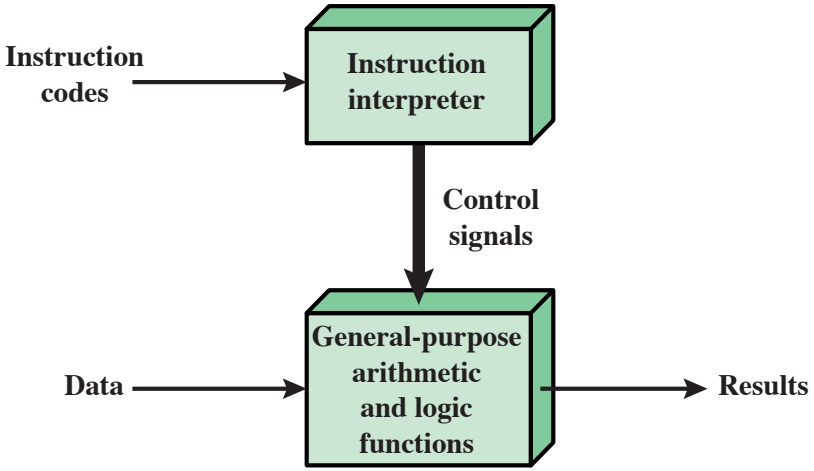
IEC

Hardware and Software Approaches

Stockholm University



(a) Programming in hardware



(b) Programming in software

Software

- A sequence of codes or instructions
- Part of the hardware interprets each instruction and generates control signals
- Provide a new sequence of codes for each new program instead of rewiring the hardware

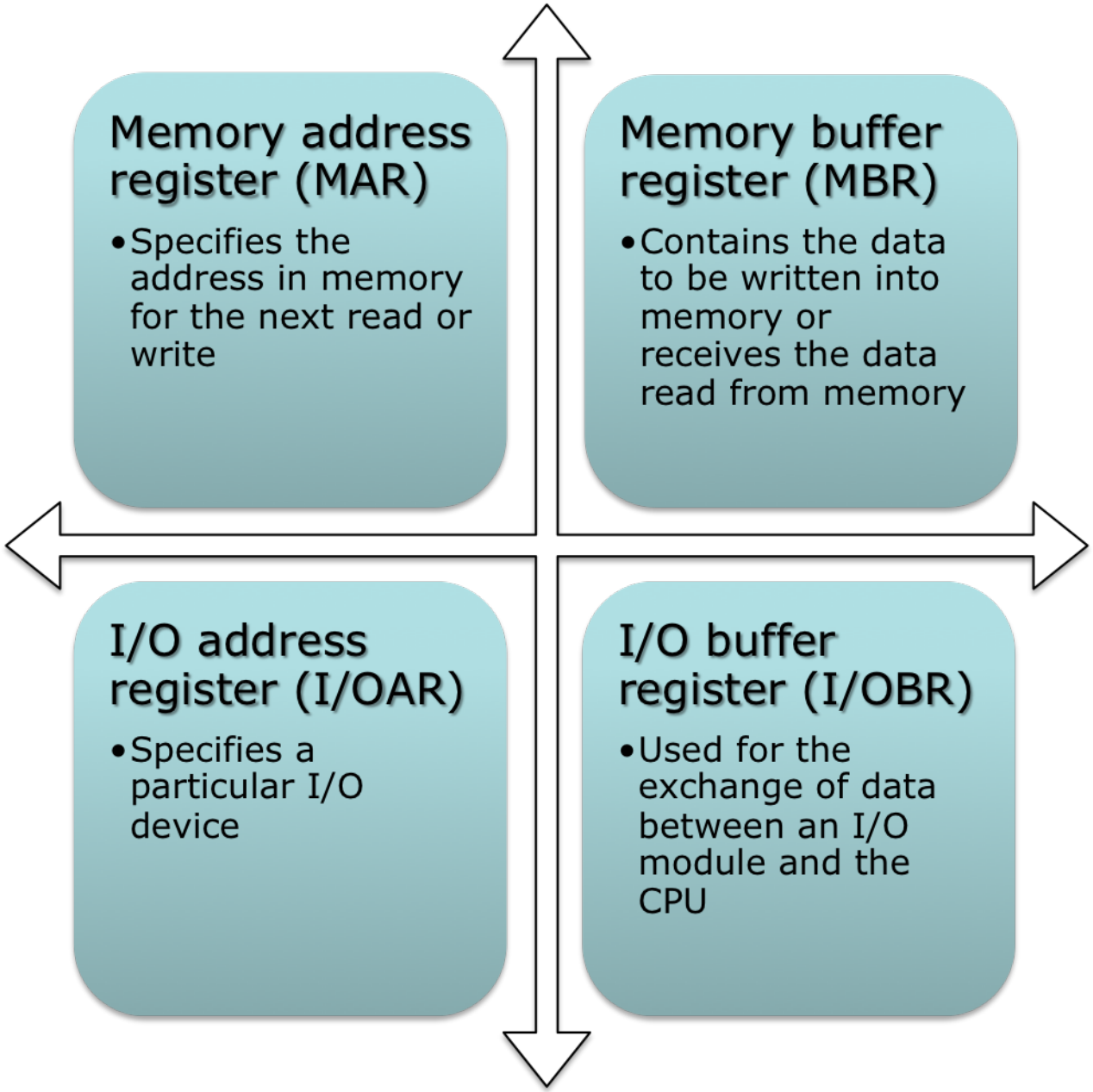
Stockholm University

Software

Major components:

- CPU
  - Instruction interpreter
  - Module of general-purpose arithmetic and logic functions
- I/O Components
  - Input module
    - Contains basic components for accepting data and instructions and converting them into an internal form of signals usable by the system
  - Output module
    - Means of reporting results

I/O Components



MEMORY

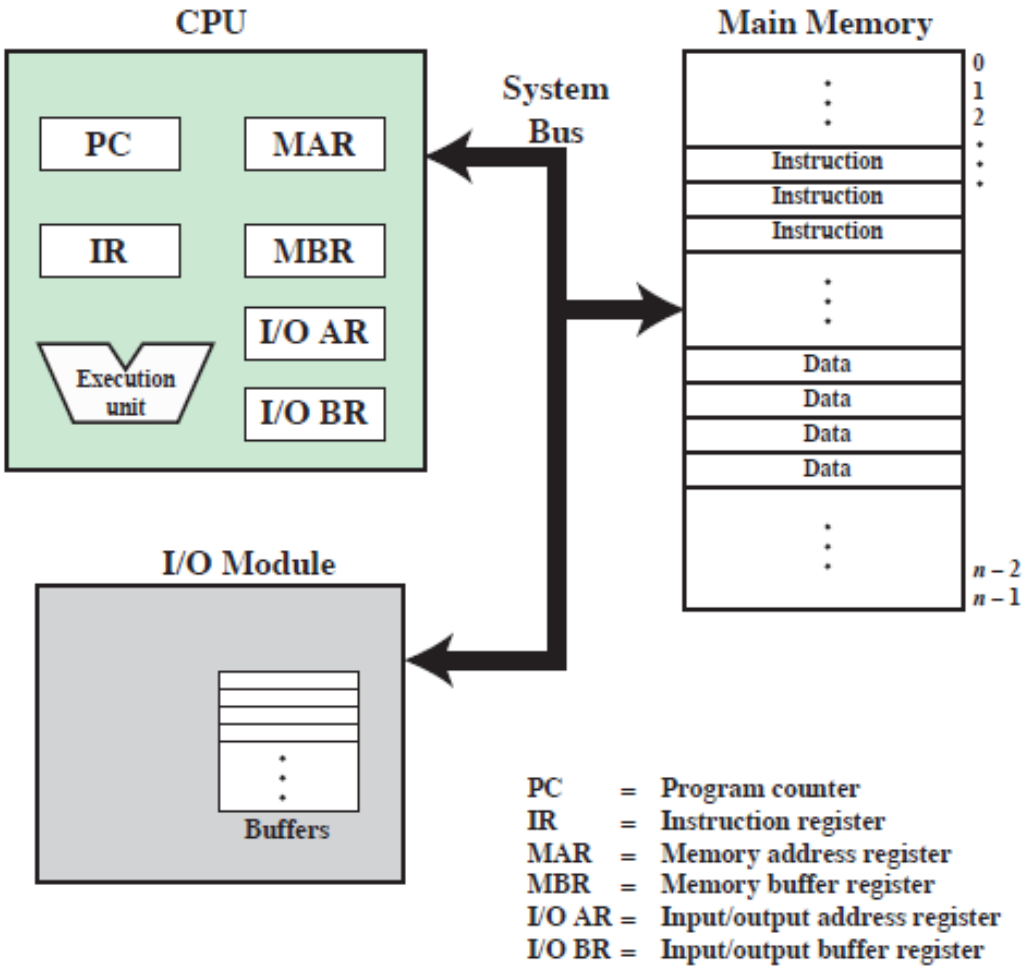
MAR

MBR

CPU Basic

Stockholm University

- Register
- ALU
- Control Unit
- System bus



CPU Basic

Stockholm University

- The computer's CPU **fetches**, **decodes**, and **executes** program instructions.
- The two principal parts of the CPU are the ***datapath*** and the ***control unit***.
  - The **datapath** consists of an **Arithmetic-Logic Unit(ALU)** and **storage units (registers)** that are interconnected by a data bus that is also connected to main memory.
  - Various CPU components perform sequenced operations according to signals provided by its control unit.

CPU Basics

Stockholm University

- **Registers** hold data that can be readily accessed by the CPU.
- The **Arithmetic-Logic Unit (ALU)** carries out logical and arithmetic operations as directed by the control unit.
- The **Control Unit** determines which actions to carry out according to the values in a *program counter register* and a *status register*.

Control and Status Register	Other Register (synliga)
• Program Counter (PC) • Instruction Regsiter (IR) • Memory Address Register(MAR) • Memory Buffer Register (MBR)	• General Register (by User) • Data Register - Only data not references • Address Register - Segmet point - Index Register - Stack point • Read only Register

Nios II Register

Stockholm University

General Register

Register	Namn	Funktion		Register	Namn	Funktion
r0	zero	0x0		r16		Callee-saved register
r1	at	Assembler temporary		r17		Callee-saved register
r2		Return value		r18		Callee-saved register
r3		Return value		r19		Callee-saved register
r4		Register argument		r20		Callee-saved register
r5		Register argument		r21		Callee-saved register
r6		Register argument		r22		Callee-saved register
r7		Register argument		r23		Callee-saved register
r8		Caller-saved register		r24	et	Exception temporary
r9		Caller-saved register		r25	bt	Breakpoint temporary
r10		Caller-saved register		r26	gp	Global pointer
r11		Caller-saved register		r27	sp	Stack pointer
r12		Caller-saved register		r28	fp	Frame pointer
r13		Caller-saved register		r29	ea	Exception return address
r14		Caller-saved register		r30	ba	Breakpoint return address
r15		Caller-saved register		r31	ra	Return address

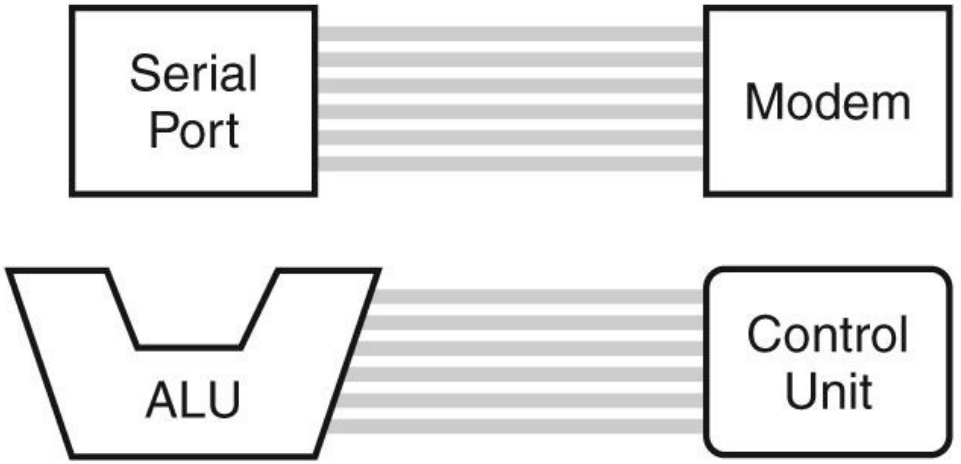
Control Register

Register	Namn
0	status
1	estatus
2	bstatus
3	ienable
4	ipending
5	cpuid
6	RESERVED
7	exception
8	pteaddr
9	tlbacc
10	tlbmisc
11	RESERVED
12	badaddr
13	config
14	mpubase
15	mpuacc

The BUS

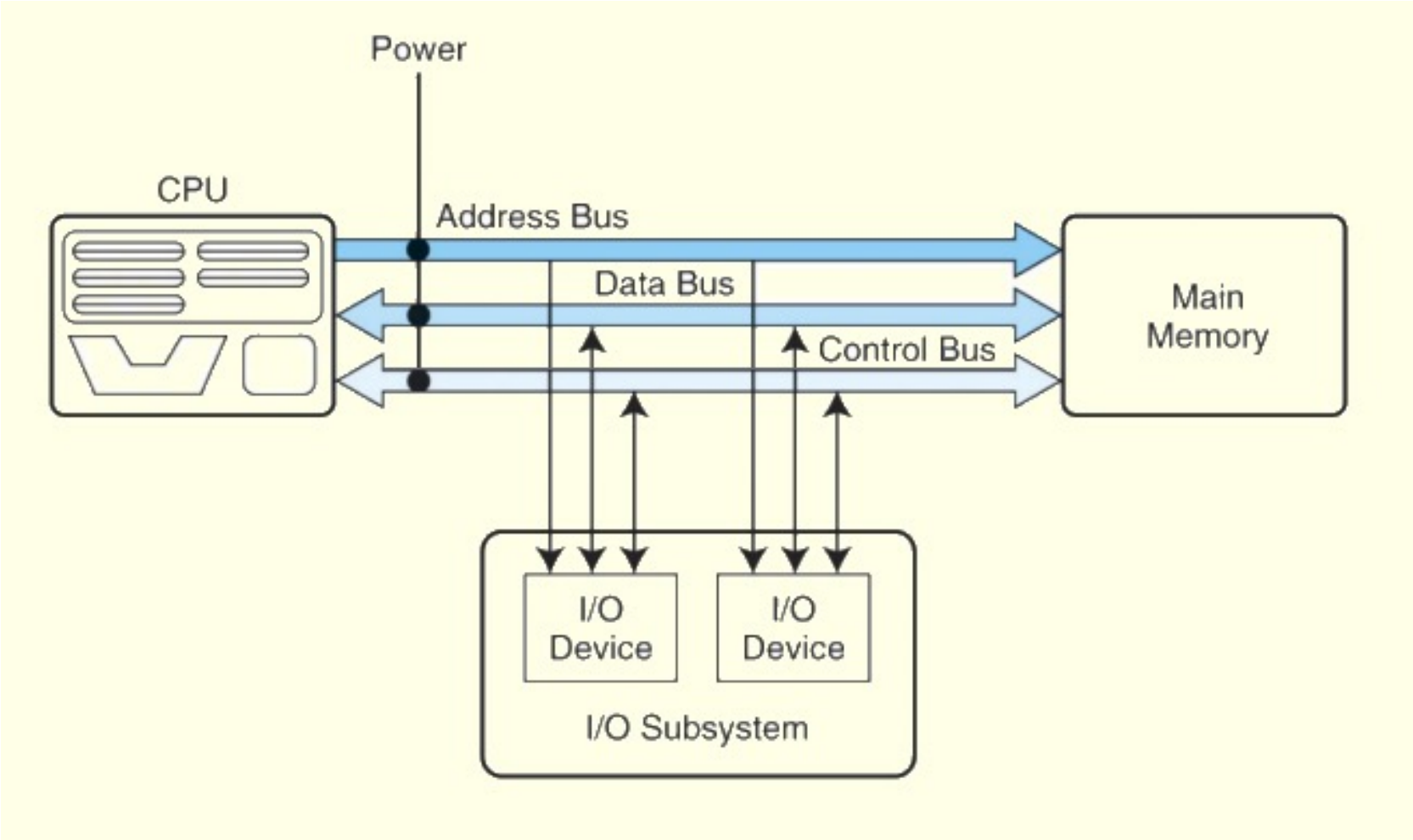
- The CPU shares data with other system components by way of a data bus.
  - A bus is a set of wires that simultaneously convey a single bit along each line.
- Two types of buses are commonly found in computer systems: *point-to-point*, and *multipoint* buses.

These are point-to-point buses:

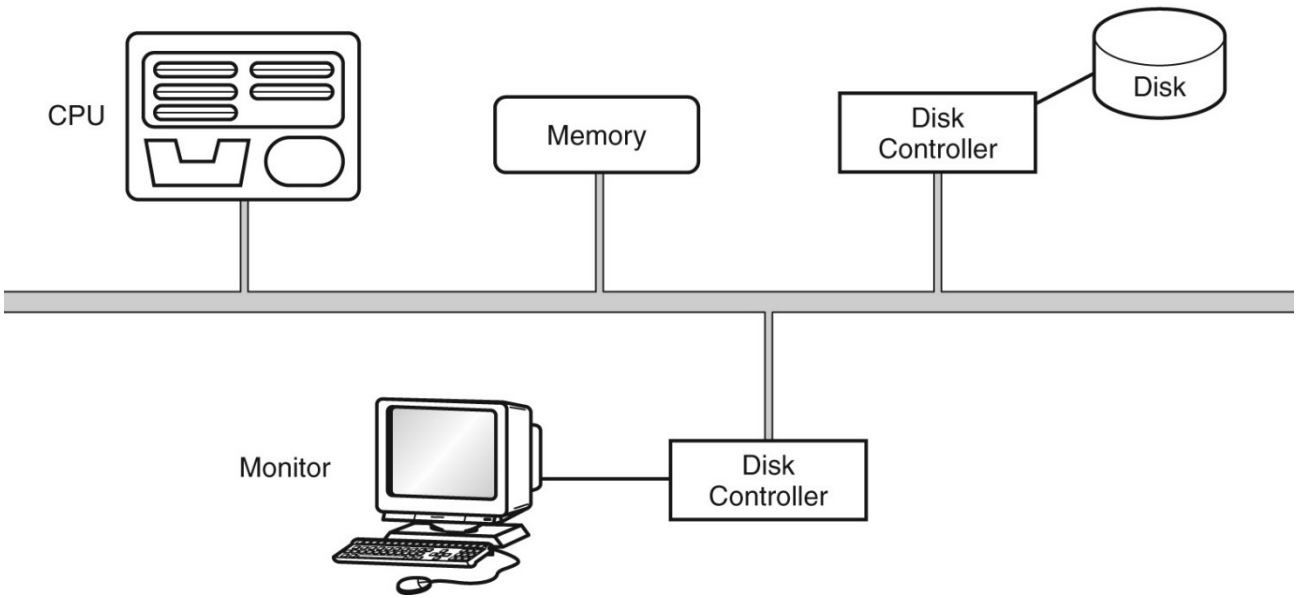


Stockholm University

- Buses consist of **data lines**, **control lines**, and **address lines**.
- While the ***data lines*** convey bits from one device to another, ***control lines*** determine the direction of data flow, and when each device can access the bus.
- ***Address lines*** determine the location of the source or destination of the data.


Power
CPU
Address Bus
Data Bus
Control Bus
Main Memory
I/O Device
I/O Device
I/O Subsystem

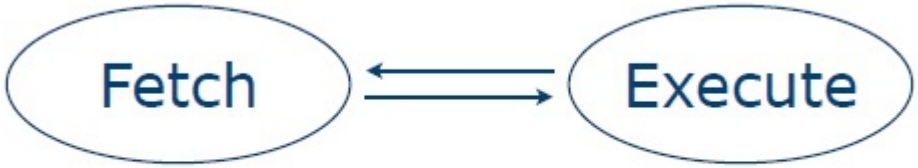
Multipoint bus



Because a multipoint bus is a shared resource, access to it is controlled through protocols, which are built into the hardware.

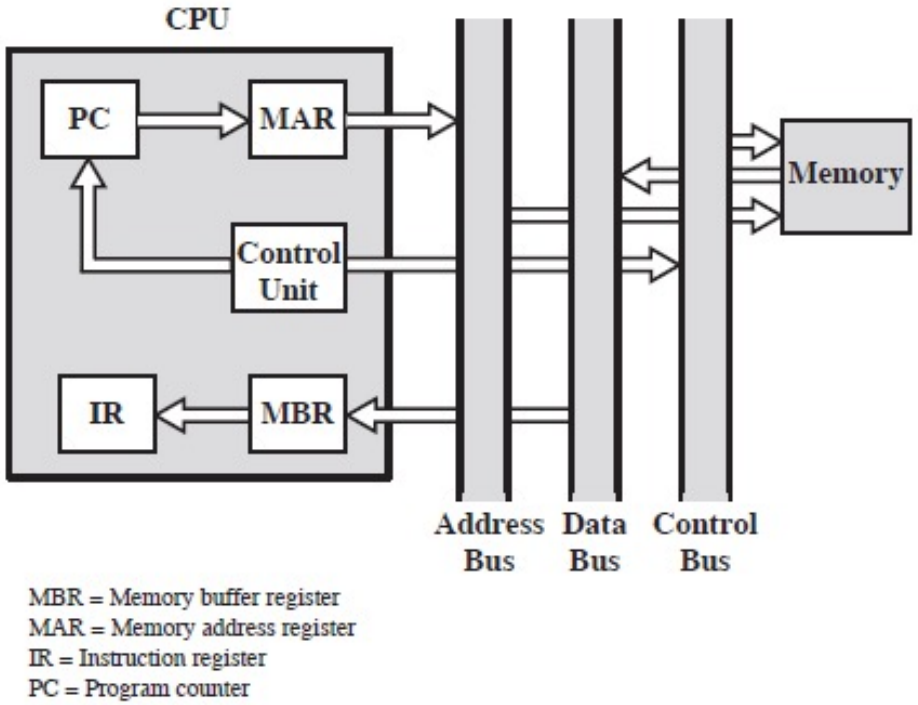
Stockholm University

CPU *fetches*, *decodes*, and *executes* program instructions



Fetch Cykel

- The value **moves** from Program Counter to MAR
- Control Unit ask for memory reading.
- The memory address reads to MBR then to IR.



Execute Cykel

Stockholm University

- Complex compare to Fetch
- Control instruction **OPeration** cod
- It is able to:
  - Move data between registers
  - Read / write of memory or I/O.
  - Aritmetic / logic caculation

Clocks

Stockholm University

- Every computer contains at least one clock that synchronizes the activities of its components.
- A fixed number of clock cycles are required to carry out each data movement or computational operation.
- The clock frequency, measured in megahertz or gigahertz, determines the speed with which all operations are carried out.
- Clock cycle time is the reciprocal of clock frequency.
  - *An 800 MHz clock has a cycle time of 1.25 ns*.

-> Clocks

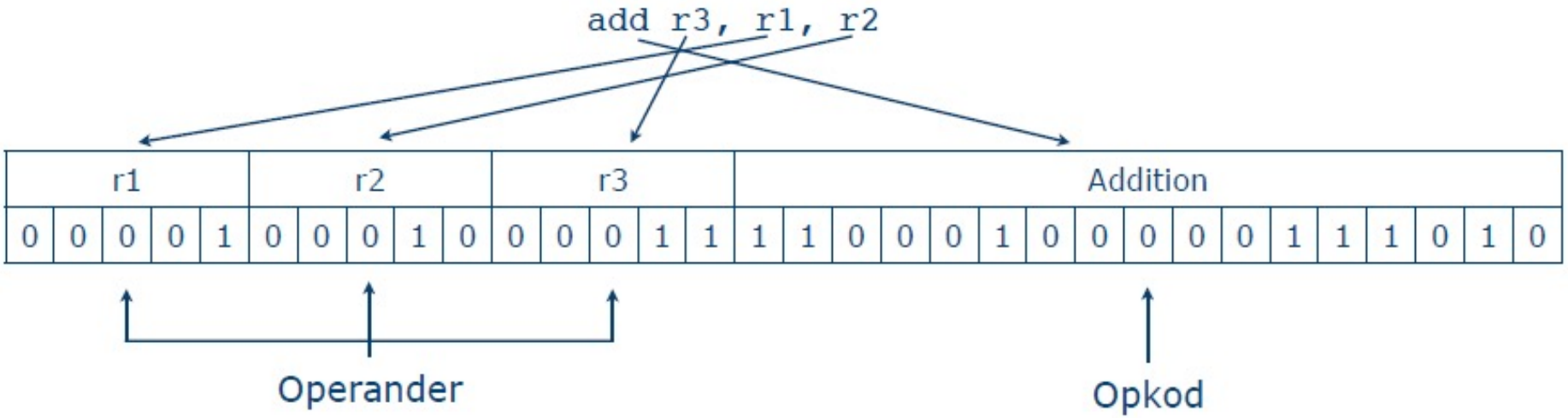
Stockholm University

- Clock speed should not be confused with **CPU performance.**
- The CPU time required to run a program is given by the general performance equation:

$$\text{CPU Time} = \frac{\text{seconds}}{\text{program}} = \frac{\text{instructions}}{\text{program}} \times \frac{\text{avg. cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

  - We see that we can improve CPU throughput when we **reduce** the number of instructions in a program, **reduce** the number of cycles per instruction, or **reduce** the number of nanoseconds per clock cycle.

Processorns operation på NIOS II



Operand format

- Register
- Immediaste
- Primärminne
- I/O

Registeroperander

add r3,r1,r2

Immediateoperander

addi r3,r1,5

Kategorier av Instruktioner

- Processor- minne
- Processor- I/O
- Aritmetiska och logiska operationer
- Flödeskontroll

Operationer

Stockholm University

Aritmetic

add
addi
sub
subi
mul
muli
div
divi

Logic

and
andi
or
ori
xor
xori
nor
...

Data

mov
movi
movia
ldb
ldw
stb
stw
...

Flow control

br
beq
blt
jmp
jmpi
call
ret
...

Nios II Instruktionsuppsättning: http://www.altera.com/literature/hb/nios2/n2cpu_nii51017.pdf

Exempel

Stockholm University

0	1	1	2	3	5	8	13	21	34	...

```
int i = 0;
int j = 1;
while (true) {
      int k = i + j;
      i = j;
      j = k;
}
```

Java-syntax

```
        movi r2, 0
        movi r3, 1
loop:   add r4, r2, r3
        mov r2, r3
        mov r3, r4
        br loop
```

NIOS II Assembly-syntax

Kontroll Instruktioner

- Branch
  - Fortsätter exekveringen på en angiven minnesadress
  - Kan kombineras med villkor
- Jump
  - Flytta exekveringen ett angivet antal rader.
  - Kombineras inte med vilkort
- Skip
  - Hoppar över nästkommande instrucktion
  - Kombineras med vilkor
  - Finns **inte** på Altera Nios II

Branches i NIOS II

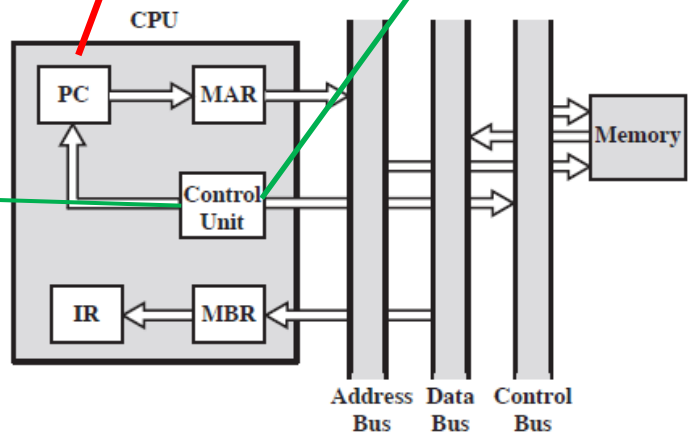
Stockholm University

br label	(Unconditional) branch to label
beq rA, rB, label	Branch if (rA == rB)
bne rA, rB, label	Branch if (rA != rB)
bge rA, rB, label	Branch if (rA >= rB)
bgt rA, rB, label	Branch if (rA > rB)
ble rA, rB, label	Branch if (rA <= rB)
blt rA, rB, label	Branch if (rA < rB)

Assemblering

```
        movi r2, 0
        movi r3, 1
loop:   add r4, r2, r3
        mov r2, r3
        mov r3, r4
        br loop
```

Adress	Instruktion				
2068	00000	00010	0000000000000000		000100
2072	00000	00011	0000000000000001		000100
2076	00010	00011	000100	11000100000	110001
2080	00010	00000	00011	11000100000	111010
2084	00011	00000	00100	11000100000	111010
2088	00000	00000	0000100000011100		000110



MBR = Memory buffer register
MAR = Memory address register
IR = Instruction register
PC = Program counter

Stockholm University

Exekvering av ett program

Minne

300	1 9 4 0
301	5 9 4 1
302	2 9 4 1
[...]	[...]
940	0 0 0 3
941	0 0 0 2

CPU-register

3 0 0	PC
	AC
1 9 4 0	IR

0001 = Hämta från minnet till AC
0010 = Spara AC till minnet
0101 = Addera AC med minnet

Stockholm University

Exekvering av ett program

Minne

300	1 9 4 0
301	5 9 4 1
302	2 9 4 1
[...]	[...]
940	0 0 0 3
941	0 0 0 2

CPU-register

3 0 1	PC
0 0 0 3	AC
1 9 4 0	IR

0001 = Hämta från minnet till AC
0010 = Spara AC till minnet
0101 = Addera AC med minnet

Stockholm University

Exekvering av ett program

Minne

300	1 9 4 0
301	5 9 4 1
302	2 9 4 1
[...]	[...]
940	0 0 0 3
941	0 0 0 2

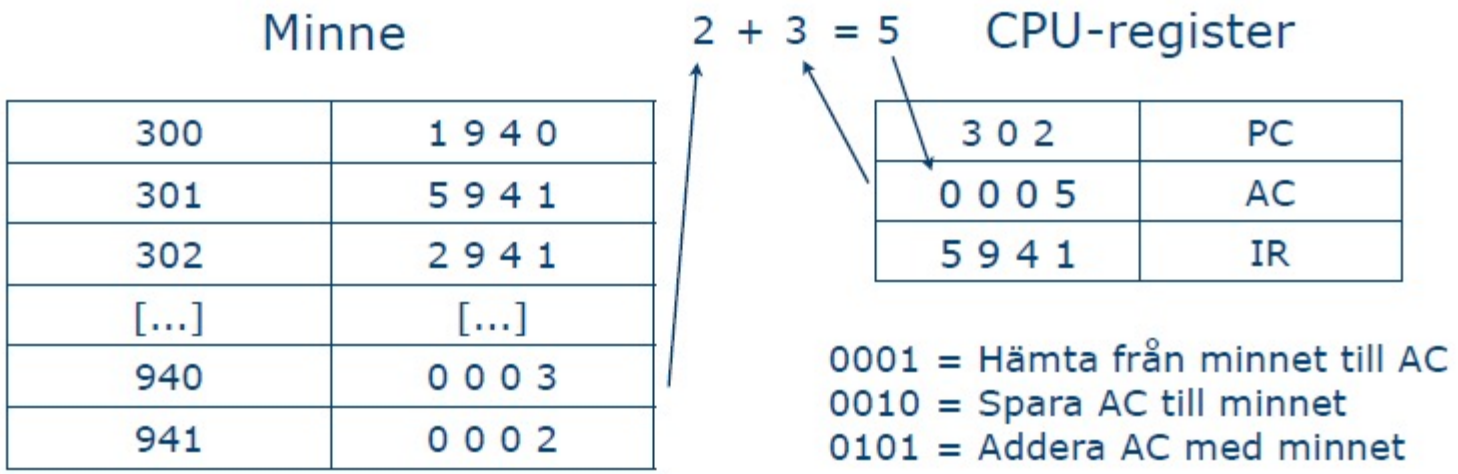
CPU-register

3 0 1	PC
0 0 0 3	AC
5 9 4 1	IR

0001 = Hämta från minnet till AC
0010 = Spara AC till minnet
0101 = Addera AC med minnet

Stockholm University

Exekvering av ett program



Stockholm University

Exekvering av ett program

Minne

300	1 9 4 0
301	5 9 4 1
302	2 9 4 1
[...]	[...]
940	0 0 0 3
941	0 0 0 2

CPU-register

3 0 2	PC
0 0 0 5	AC
2 9 4 1	IR

0001 = Hämta från minnet till AC
0010 = Spara AC till minnet
0101 = Addera AC med minnet

Stockholm University

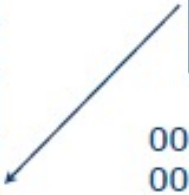
Exekvering av ett program

Minne

300	1 9 4 0
301	5 9 4 1
302	2 9 4 1
[...]	[...]
940	0 0 0 3
941	0 0 0 5

CPU-register

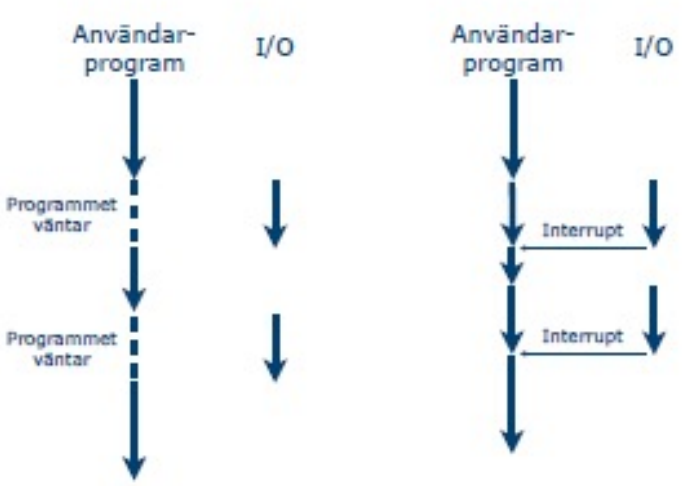
3 0 3	PC
0 0 0 5	AC
2 9 4 1	IR



0001 = Hämta från minnet till AC
0010 = Spara AC till minnet
0101 = Addera AC med minnet

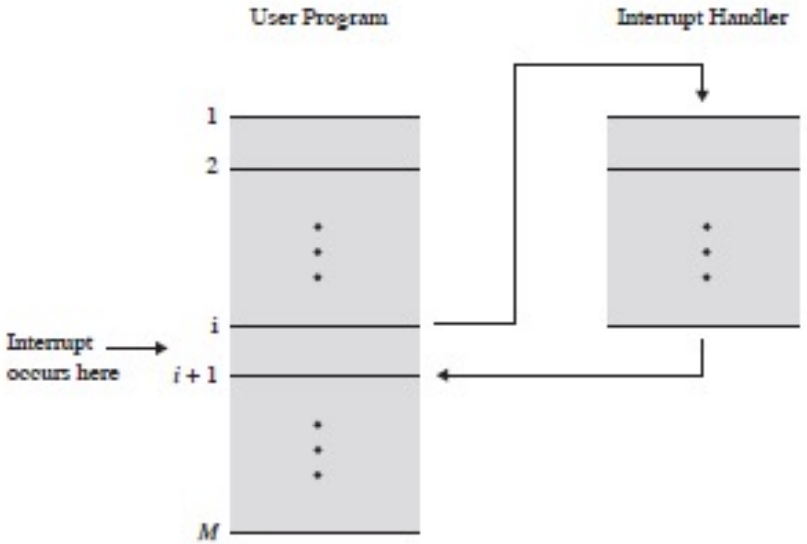
Interupts

- Ibland är det smidit att kunna avbryta fect/execute-cykeln
  - Timer
  - I/O
  - Om fel inträffar



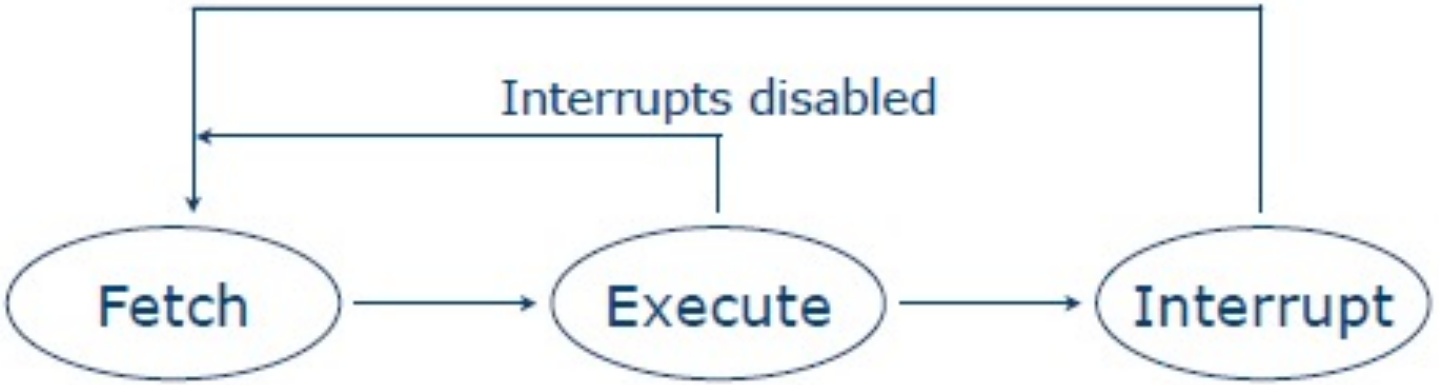
Hur sker en interrupt?

- En enhet som vill att processorn ska göra ett avbrott antecknar det i ett register.
- Processorn kontrollerar efter execute-cykeln om avbrott ska göras.
- Om avbrott ska göras ändras programräknarens värde till adressen för avbrottshanteraren.



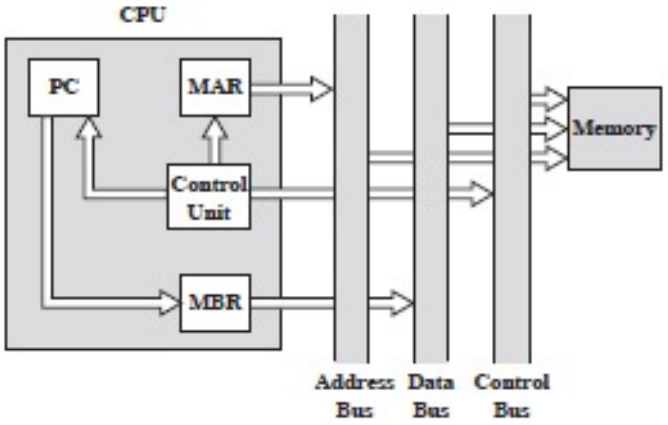
Stockholm University

Instrucktionscykeln med Interrupts

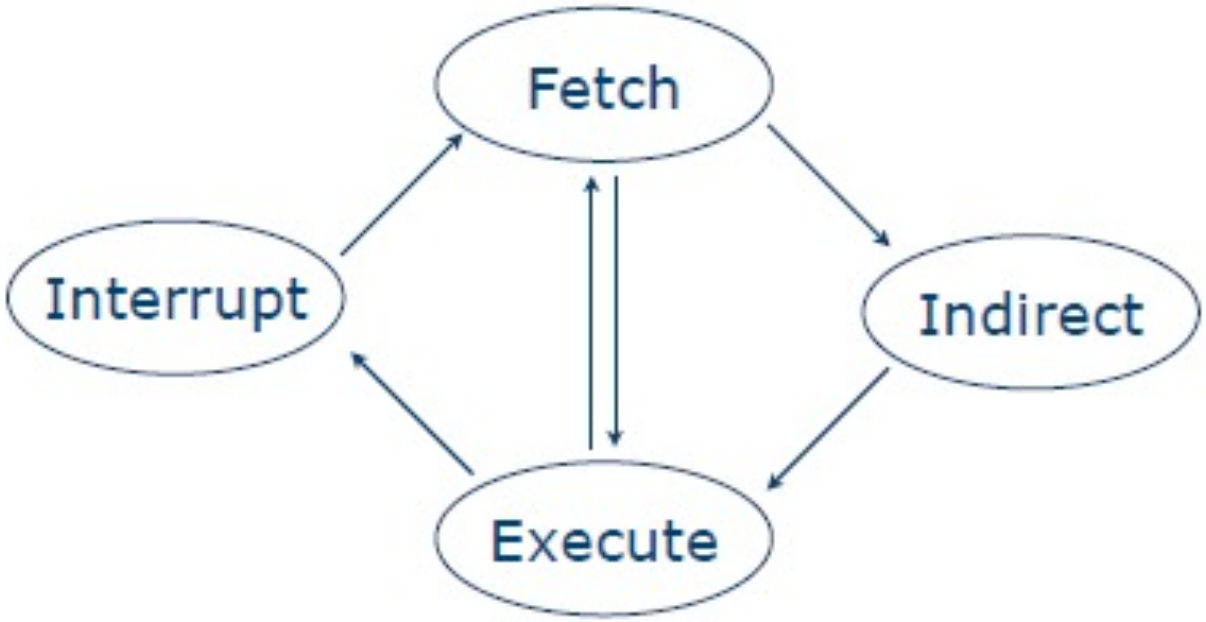


Interruptcykeln

- Innehållet i programräknaren flyttas till MBR och skrivs till minnet.
- Adressen till avbrottshanteraren flyttas till programräknaren.

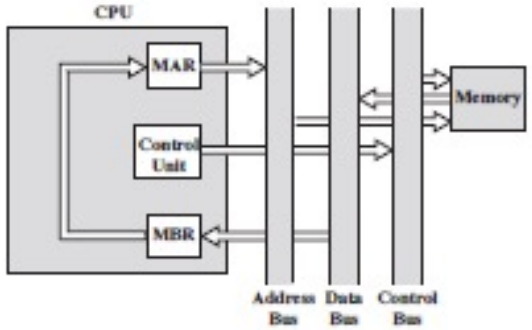


Instrucktionscykeln med Indirect-steg

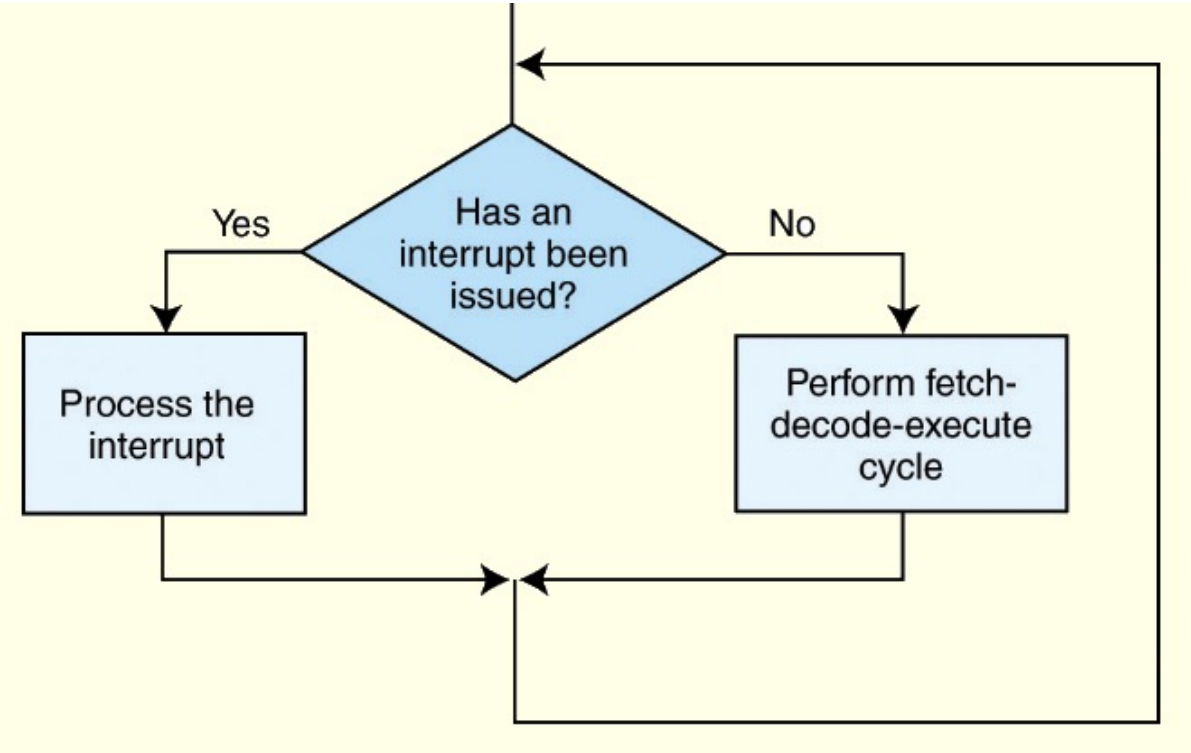


Indirectcykeln

- Minnesreferensen flyttas från MBR till MAR.
- Kontrollenheten begär en minnesläsning.
- Minnesadressen läses in till MBR.

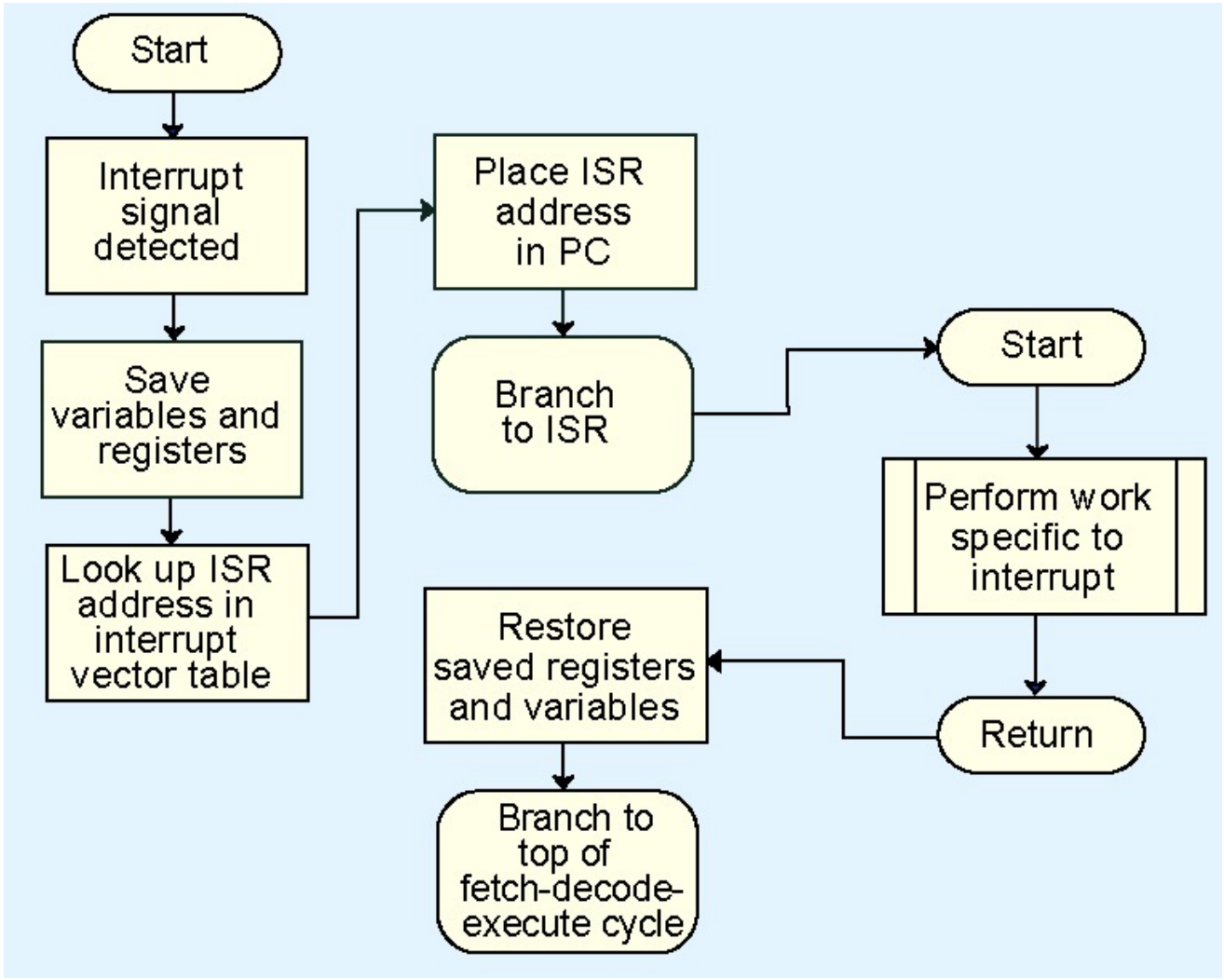


Interrupt processing involves adding another step to the *fetch-decode-execute* cycle as shown below



Interrupting the fetch-decode-execute cycle

Stockholm University



Maskable & NonMaskable Interrupts

Stockholm University

- For general-purpose systems, it is common to disable all interrupts during the time in which an interrupt is being processed.
  - Typically, this is achieved by setting a bit in the flags register.
- Interrupts that are ignored in this case are called *maskable*.
- *Nonmaskable* interrupts are those interrupts that must be processed in order to keep the system in a stable condition.

Summary

- Boolean Algebra & Digital Logic
- The Bus
- CPU Basic
- CPU: executes program instructions
- Processor operation: NIOS II
- Interrupts
- Maskable & NonMaskable Interrupts