Laborator 3
Arhitectura software

Obiective

- Noțiuni arhitectură software
- Modelul AUTOSAR
- Arhitectura SW a produsului

Cuprins

Ce este arhitectura software?

Arhitectura de software se referă la structurile fundamentale ale unui sistem software; reprezintă disciplina care crează și documentează astfel de structuri. Fiecare structură cuprinde elemente de software, relațiile dintre ele, și proprietăți ale elementelor și relațiilor dintre elemente împreună cu justificarea pentru introducerea și configurația fiecărui element.

Arhitectura unui sistem software este o metaforă, analog cu arhitectura unei clădiri.

Arhitectura software este despre a face alegeri structurale fundamentale, care sunt costisitoare pentru a fi schimbate ulterior.

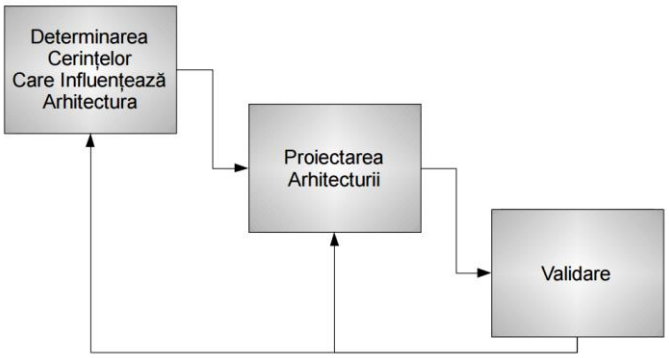
Arhitectura software facilitează documentarea, comunicarea între părtile interesate, surprinde decizii cu privire la proiectarea arhitecturii și permite reutilizarea componentelor de proiectare între proiecte.

Responsabilitatea unui arhitect software nu constă doar din activitatea de proiectare. Un arhitect software trebuie să:

- **lucreze cu echipa care stabilește cerințele aplicației:** în acest context rolul arhitectului este acela de a ajuta la adunarea cerințelor și înțelegerea nevoilor sistemului ca ansamblu și de a asigura faptul că indicatorii de calitate sunt expliciți și bine înțeleși;
- **lucreze cu toți cei care vor interacționa cu aplicația**: în acest context arhitectul are rolul de pivot, el trebuie să se asigure că toate nevoile celor care vor interacționa cu aplicația sunt înțelese și încorporate în proiect. De exemplu, pe lângă cerințele care se referă la funcționalitate, administratorul de rețea poate să ceară ca aplicația să fie ușor de instalat, configurat și actualizat;
- **conducă echipa tehnică de proiectare**: definirea arhitecturii unei aplicații este o activitate de proiectare. Arhitectul conduce echipa de proiectare, compusă din proiectanți de sistem și conducători de echipe de programatori, pentru a realiza schița arhitecturii;
- **lucreze cu managerul de proiect:** arhitectul trebuie să ajute la planificarea proiectului, estimarea și alocarea taskurilor.

Procesul de proiectare a arhitecturii unui sistem software complex constă din parcurgerea iterativă a trei etape:

1. **Determinarea cerințelor care influențează arhitectura:** presupune crearea unui model al cerințelor care va dirija proiectarea arhitecturii.
2. **Proiectarea arhitecturii:** definirea structurii și a responsabilităților componentelor care formează arhitectura.
3. **Validarea – testarea arhitecturii:** de obicei se realizează parcurgând lista de cerințe și eventualele cerințe ulterioare și verificarea faptului că arhitectura proiectată în etapa anterioară permite implementarea cerințelor.



Determinarea cerințelor care influențează arhitectura**.** Înainte de a proiecta arhitectura unui sistem software este important să se obțină o imagine clară asupra cerințelor care influențează arhitectura. De obicei aceste cerințe sunt cerințe non-funcționale care se referă la calitatea sistemului software. Procesul de identificare a cerințelor care afectează arhitectura are două tipuri de intrări: pe de o parte arhitectul trebuie să analizeze cerințele funcționale, iar pe de altă parte el trebuie să țină cont și de cerințele venite din partea celor care vor interacționa cu aplicația.

Proiectarea arhitecturii. Reprezintă cea mai importantă acțiune întreprinsă de către arhitect. /*Un document de cerințe foarte bine structurate, respectiv o comunicare bună cu restul echipelor implicate în proiect nu însemnă nimic dacă se proiectează o arhitectură slabă*/. Etapa de proiectare a arhitecturii are ca și intrări cerințele obținute în etapa anterioară iar ca rezultat se obține un document care descrie arhitectura sistemului software. Proiectarea arhitecturii se realizează în doi pași: primul pas se referă la alegerea unei strategii globale, iar al doilea constă din specificarea componentelor individuale și a rolului pe care fiecare componentă îl joacă în arhitectura globală.

Alegerea strategiei globale**.** De obicei alegerea strategiei globale se bazează pe un număr restrâns de modele predefinite și bine înțelese. Așadar primul pas din cadrul etapei de proiectare constă în alegerea unui cadru care să satisfacă cerințele cheie ale sistemului.

Definirea componentelor. După ce a fost definit scheletul arhitecturii prin selectarea unui model, următorul pas constă din definirea principalelor componente. Pentru aceasta trebuie să se:

- identifice principalele componente ale sistemului, precum și modul în care ele sunt integrate în scheletul arhitecturii;
- identifice interfața și serviciile disponibile pentru fiecare componentă în parte;
- identifice responsabilitățile componentelor;
- identifice dependențele între componente;
- identifice bucățile din arhitectură care pot fi refolosite în alte proiecte similare.

Validarea arhitecturii**.** Scopul etapei de validare este acela de a verifica faptul că arhitectura proiectată este potrivită pentru sistemul software care urmează să fie dezvoltat. Principala dificultate în ceea ce privește validarea unei arhitecturi constă în faptul că în acest moment nu există un produs software “fizic” care să poată fi executat și testat. Există două metode prin care se poate valida arhitectura unui sistem software: testarea manuală utilizând scenarii, respectiv validare prin construirea unui prototip.

Utilizarea scenariilor. Utilizarea scenariilor pentru validarea arhitecturii unui sistem software presupune definirea unor stimuli care să aibă efect asupra arhitecturii. După care se face o analiză pentru a se determina care va fi răspunsul arhitecturii la un astfel de scenariu. Dacă răspunsul este cel dorit atunci se consideră că scenariul este satisfăcut de arhitectură. Dacă răspunsul nu este cel dorit sau este greu de calificat atunci s-a descoperit o zonă de risc în arhitectură. Se pot imagina scenarii care să evalueze oricare din cerințele sistemului.

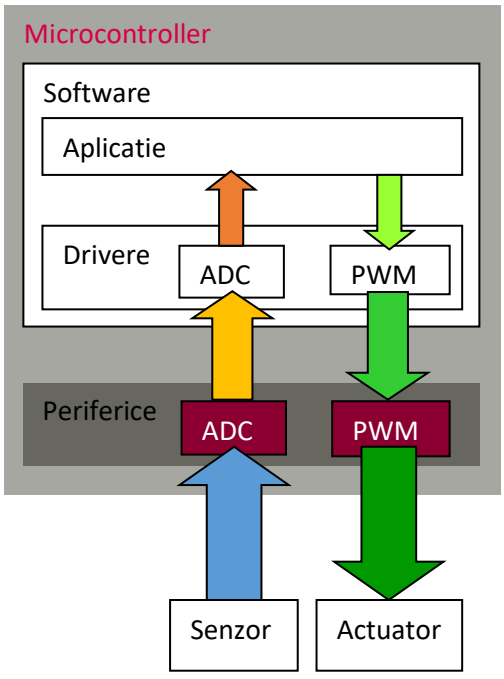
Crearea unui prototip**.** Deși scenariile sunt tehnici foarte utile în vederea testării unei arhitecturi, nu întotdeauna verificarea unui scenariu se poate face doar prin analiza arhitecturii, de aceea în anumite situații se recurge la construirea unui prototip care să permită verificarea anumitor scenarii. Un prototip se poate construi din două motive:

- *Proof-of-concept*: verifică dacă se poate implementa arhitectura proiectată astfel încât să satisfacă cerințele;
- *Proof-of-technology*: verifică faptul că tehnologia middleware selectată se comportă așa cum se așteaptă.

Odată ce prototipul a fost implementat și testat răspunsul arhitecturii la stimulii prevăzuți în scenarii se poate obține cu un înalt grad de certitudine. Deși un prototip poate fi foarte util în ceea ce privește validarea unei arhitecturi, trebuie avut grijă ca dezvoltarea unui astfel de prototip să nu dureze prea mult. De obicei un prototip este abandonat după ce arhitectura a fost validată, de aceea dezvoltarea unui prototip nu trebuie să dureze mai mult de câteva zile, cel mult 1-2 săptămâni.

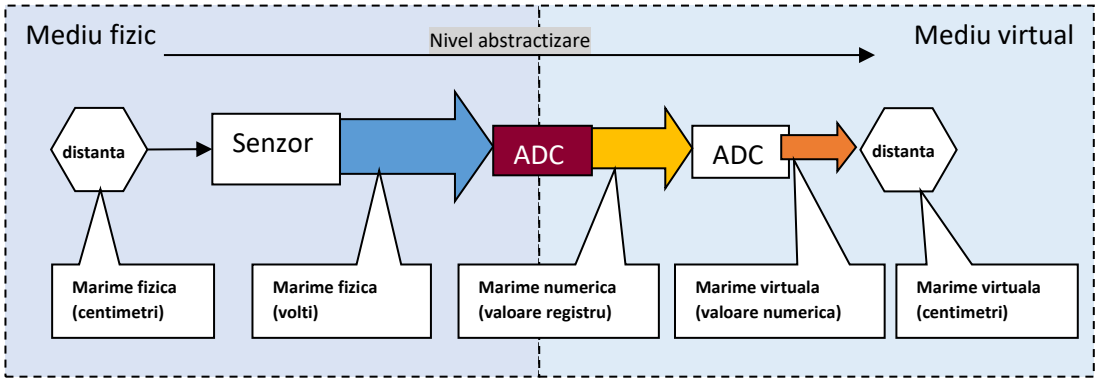
Modelarea unui process controlat în embedded

Calea informației într-un sistem embedded:

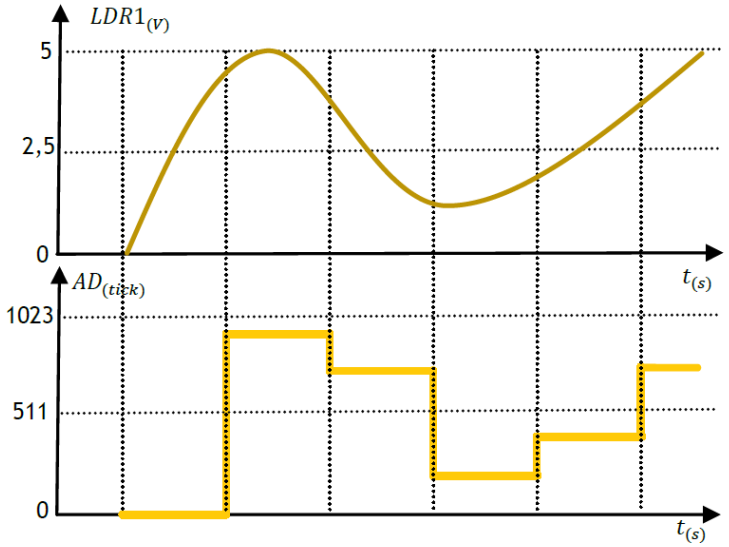


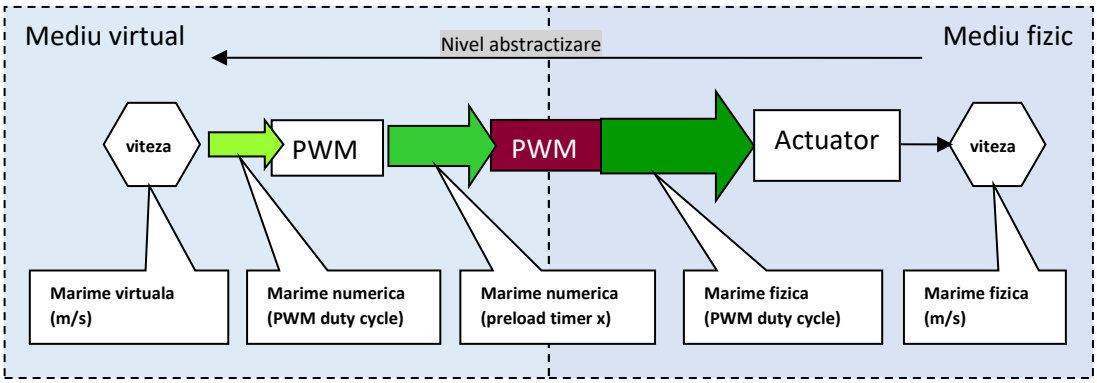
Software-ul de pe un microcontroler se compune din 2 părți:

- parte de cod care gestionează perifericele existente în microcontroller și care adițional transformă valorile numerice raportate de periferice în valori virtuale echivalente valorilor fizice din mediul înconjurător;
- parte de cod care ia decizii (controlează) în funcție de valorile măsurate (de intrare) și care reacționează prin modificarea valorilor de comandă (de ieșire).

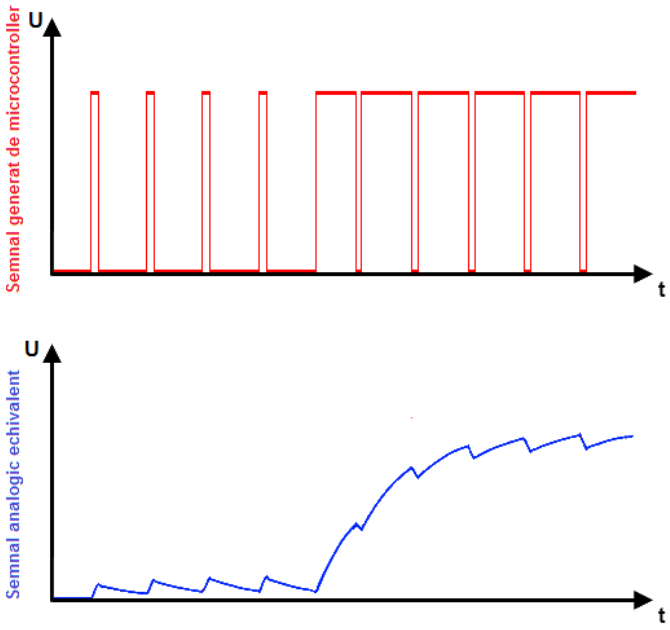


ADC (Convertor Analog catre Digital) este un periferic specializat capabil să transforme semnalul de tensiune analogic într-o valoare digitală.





PWM (Modulator de Latime a Pulsului) este un periferic specializat capabil să genereze un semnal periodic şi digital cu factor de umplere configurabil.



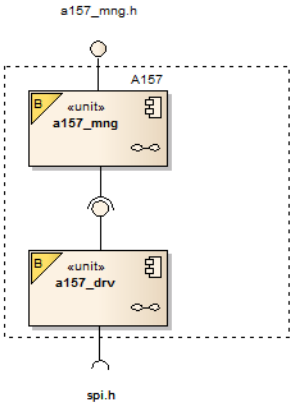
Noțiuni arhitectura software

SW Unit- pereche de fișiere .c si .h care implementează un set de funcționalități specifice.

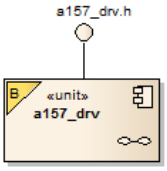
SW Component- grup de SW unit-uri care lucrează împreună pentru a oferi un set de funcționalități complexe.

SW Layer- grup de SW componente care realizează un anumit nivel de abstractizare.

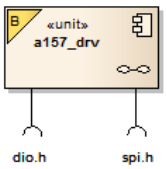
Interfete- legatura dintre 2 SW unit-uri – modalitatea de comunicare (schimb date) între unit-uri.



Interfete oferite- reprezintă colectia de variabile şi funcții pe care un SW unit le oferă spre folosința celorlalte unit-uri.

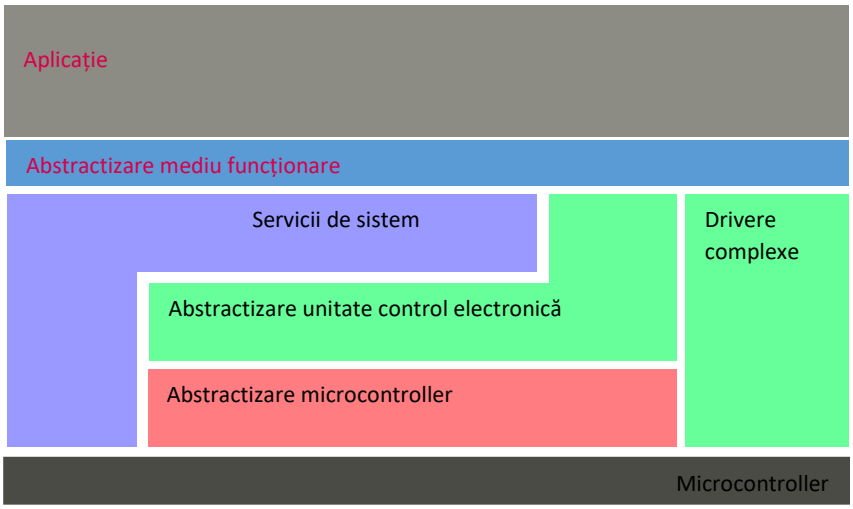


Interfete necesare- reprezintă colectia de variabile şi funcții de care un SW unit are nevoie pentru a implementa propriile funcționalități.



Modelul AUTOSAR

AUTOSAR (Automotive Open System ARchitecture) reprezintă un parteneriat de dezvoltare la nivel global între producătorii de autovehicule, furnizorii și alte companii din domeniul industriei de electronice, de semiconductoare și de software.



Componentele SW din nivelul servicii de sistem oferă:

- sistemul de operare;
- acces către mediul de memorie volatilă și non-volatile;
- acces către componenta de diagnoză și stocare a erorilor;
- starea curentă a sistemului;
- monitorizarea temporală și de algoritm al sistemului.

Componentele SW din nivelul de abstractizare al microcontrolerului oferă independență față de tipul microcontrolerului, prin implementarea unor interfețe standard pentru componentele superioare.

Componentele SW din nivelul de abstractizare unitate control electronică facilitează transformările din semnalele recepționate în semnale fizice care pot fi interpretate de către aplicație.

Componentele SW din nivelul de drivere complexe implementează drivere care nu sunt specificate în standardul AUTOSAR și care au access în toată partea de platformă necondiționat.

Componentele SW din nivelul de abstractizare a mediului de funcționare au rolul de a deconecta partea de SW de platformă de partea de SW de aplicație.

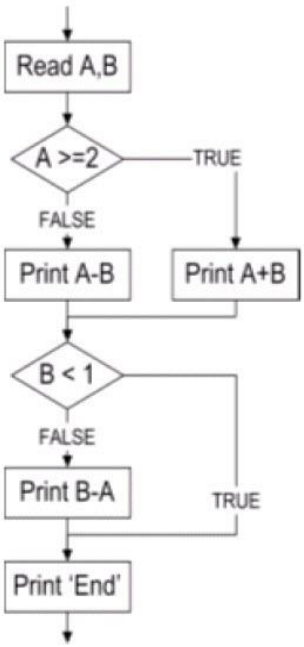
Componentele SW din nivelul de aplicație au rolul de a implementa partea funcțională a produsului.

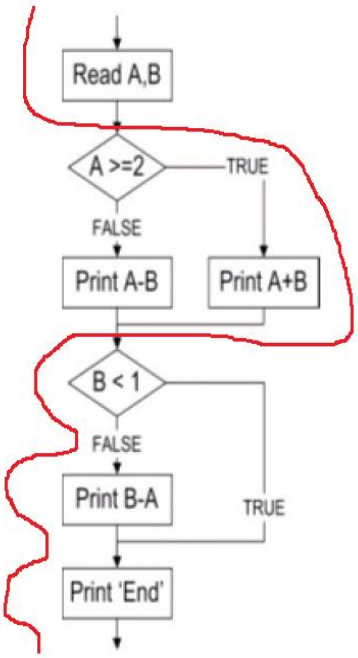
Metode de testare

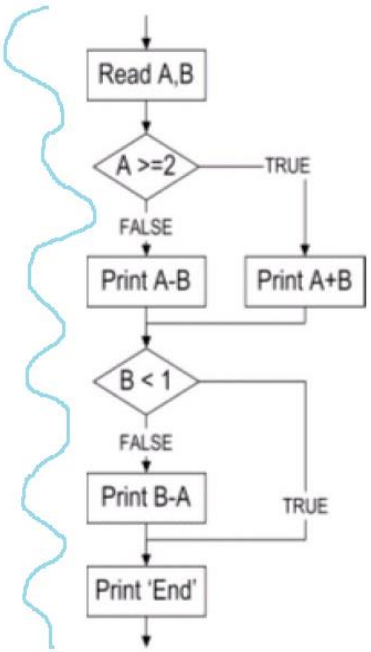
Spre deosebire de testarea black box, în cazul în care testerul dorește să se concentreze pe structura internă a codului sursă a unui sistem embedded, acesta trebuie să se axeze pe tehnicile de testare **white box:**

- **Statement coverage**
  - este folosită pentru a calcula și măsura numărul de instrucțiuni de cod care au fost executate;
  - acoperă doar condițiile adevărate;
  - este folosită pentru a identifica instrucțiunile executate și codul mort și măsoară calitatea codului.
  - formula de calcul este

$$Statement\ coverage = \frac{nr\ de\ instructiuni\ executate}{nr\ total\ de\ instructiuni} \cdot 100\%$$







A = 1, B = 2

SC = 6/7 = 85%

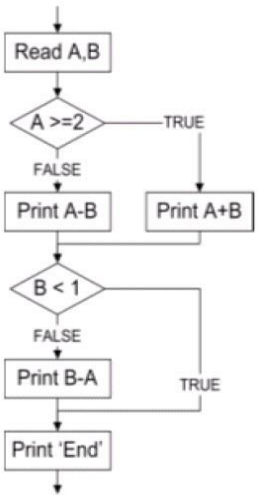
A = 2, B = 2

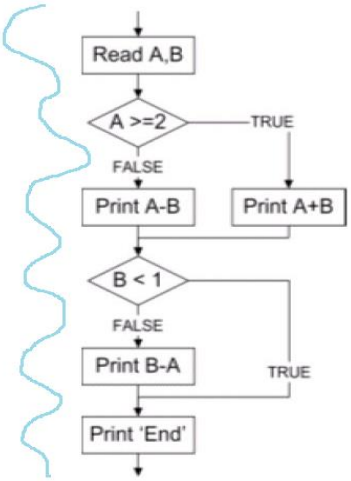
SC = 6/7 = 85%

SC_total = 100 % cu doua cazuri de test

- **Branch coverage**
  - presupune executarea fiecărei ramuri a fiecărui punct de decizie, cel puțin o dată
  - acoperă atât condițiile adevărate cât și cele false
  - este folosită la validarea parcurgerii tuturor ramurilor
  - formula de calul este

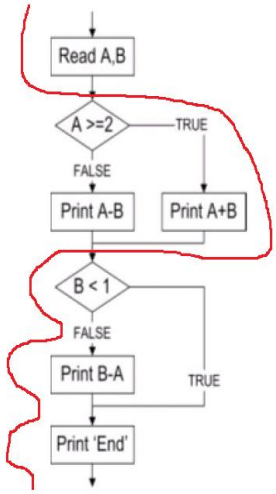
$$Branch\ coverage = \frac{nr\ deciziilor\ indeplinite}{nr\ total\ de\ decizii} \cdot 100\%$$





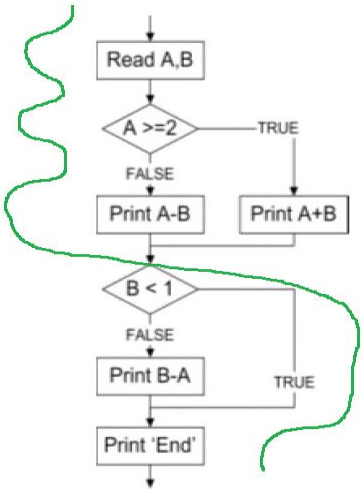
A = 1, B = 2

BC = 2/4 = 50%



A = 2, B = 2

BC = 2/4 = 50%

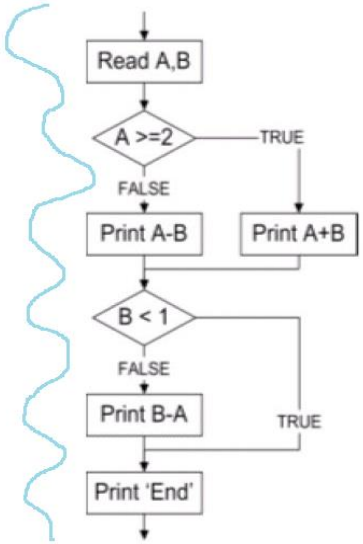


A = 1, B = 0

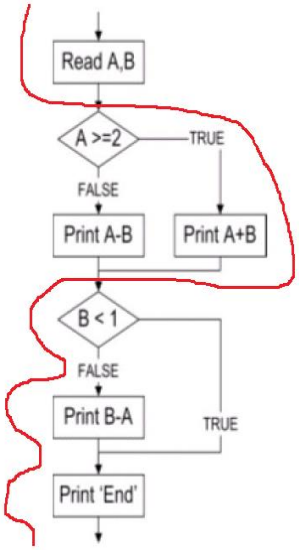
BC = 2/4 = 50%

BC_total = 100% cu trei cazuri de test

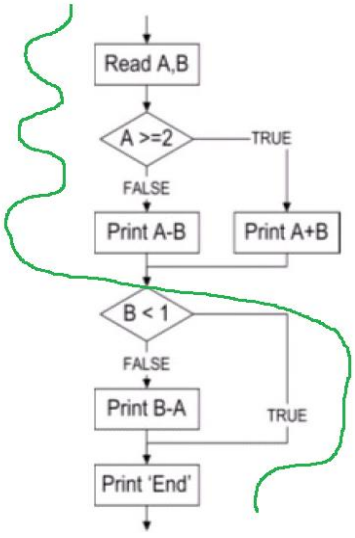
- **Path coverage** – presupune execuția tuturor combinațiilor posibile într-un program



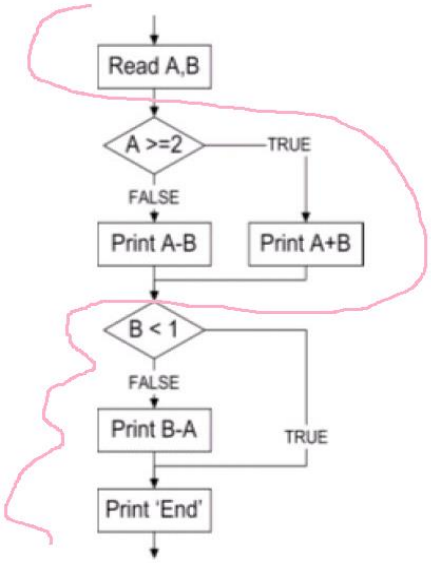
A = 1, B = 2



A = 2, B = 2



A = 1, B = 0



A = 2, B = 2

PC_total = 100% cu patru cazuri de test

Complexitatea este de 2^n, unde n = numărul de decizii.

- **Ghicitul erorilor (Error guessing)**
    - Bazându-se pe experiența acumulată, testerii se gândesc la situații în care software-ul nu ar putea face față.
    - Cazurile de teste uzuale sunt: diviziunea prin 0, fișier gol sau format gresit, caractere alfabetice în loc de numerice.
    - O abordare mai structurată presupune crearea unei liste cu probleme care pot apărea cu trecerea timpului și încercarea de a le reproduce.

De obicei, aceste metode de testare sunt făcute cu ajutorul anumitor instrumente specializate.

Q&A session

Arhitectura unui sistem embedded poate fi de multe ori greu de înțeles. Astfel, modelul AUTOSAR vine în ajutorul dezvoltatorilor prin standardizarea modului în care este construită arhitectura unui sistem din industria automotive.