CODE ACADEMY

C++ Team Project 1

ASAPML 1.0

Automated Systems for Advanced Printing of Microchip Layers

(As Soon As Possible Manufacturing Line)

CODE ACADEMY

ASAPML 1.0

ASAPML е съвкупност от **взаимосвързани и взаимодействащи си програми**, които са абстрактна симулация на **софтуерна система за автоматизация на процес за поръчка и производство на хардуерни компоненти**. Състои се от следните програми:

- **ASAPML Ω (Ohm - Order “Harmony” Manager)**

  Програма за правене на поръчки от клиенти. Клиентите имат възможността да направят поръчка на определен хардуерен компонент (давайки blueprint файл, описващ дизайна на компонента). Поръчките се записват във файл с необработени поръчки.

- **ASAPML Amp (Asset Monitoring Program)**

  Програма за **менажиране на инвентара**. За направата на даден **хардуерен компонент** са необходими множество от конкретни **електронни компоненти** (резистори, транзистори, чипове, ..). Целта на програмата за менажиране на инвентара е да **автоматизира следенето за налични ресурси** (електронни компоненти), и да **прави поръчки** за необходимите количества и видове електронни компоненти, както **и ги приема**.

- **ASAPML Bit (Bitwise Integration Transfer)**

  Програма, която **симулира работата на машина за принтиране** на хардуерни компоненти. Приема файл със заявки за принтиране/запояване/.., **принтира за определено време** и вместо да комуникира с реална машина, която принтира истински физически хардуерни компоненти, с**ъздава файлове, репрезентиращи отделни физически хардуерни компоненти** (готова продукция)

- **ASAPML Volt (Virtual Operations and Logistics Tool)**

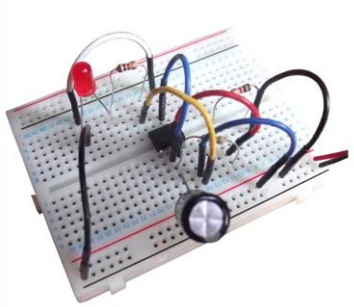
  Програма-диригент (controller, scheduler), която **оркестрира цялостния процес по ефективен начин**.
  Комуникира с останалите приложения, посредствеом файлове (в текущата версия).

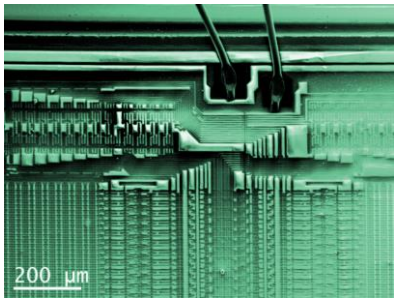
CODE ACADEMY

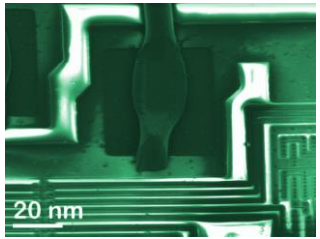
ASAPML 1.0

Проектът е абстрактна репрезентация на автоматизиран процес за принтиране на хардуерни модули.

- Вместо работа с истински машини, симулираме процеса с отделни програми и файлове.
- Вместо истински физически продукт, пишем във файл.
- Не гоним реализъм и можем да си представяме, че принтираме на каквото си искаме ниво на детайл:





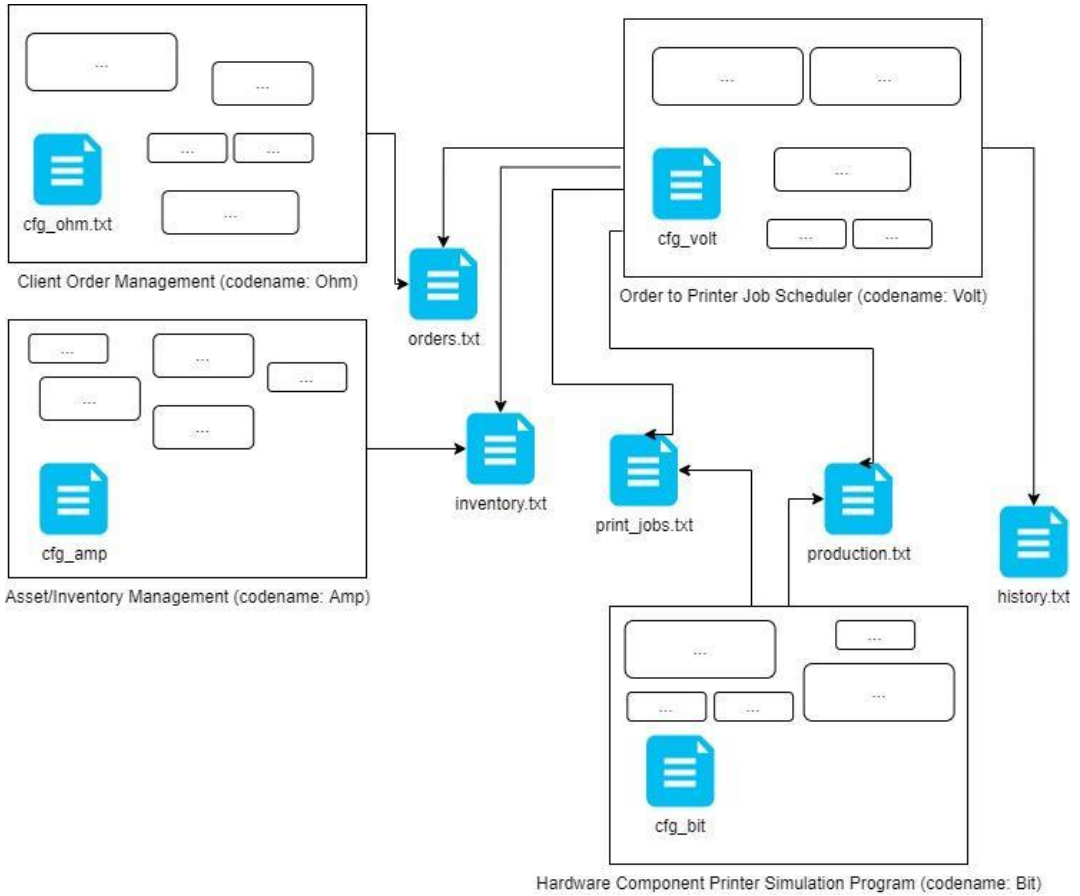


CODE ACADEMY

ASAPML 1.0

Common Use Case(s):

1. Оператор въвежда нова **клиентска заявка** в Ohm.
2. **Ohm добавя заявката** в **orders.txt**, маркирайки я “**unprocessed**”
3. **Volt** з**абелязва новата заявка в orders.txt** и **започва опит да я обработи**, понеже е марикана като “**pending**” или “**unprocessed**” (второто в случая. “**pending” заявки са с приоритет** пред “unprocessed”
4. **Volt проверява в inventory.txt за налични ресурси**, необходими за заявката:
    a. Ако **има**: добавя **нова заявка** за принтиране **в print_jobs**, **приспада ресурси от inentory.txt** и **маркира** заявката със статус: **“printing” в orders.txt.**
    b. Ако **няма**: **маркира** заявката като “**pending**” **в orders.txt**, правейки я кандидат за повторен опит за обработка на по-късен етап.
5. **Amp** от своя страна с**леди за налични ресурси** и **автоматично прави поръчки** за нови ресурси, на базата на потребителска конфигурация.
6. **Bit забелязва нови заявки за принтиране в print_jobs.txt** и започва работа по принтиране. При завършено принтиране, вместо реален физически хардуерен компонент се **създава запис в production.txt.**
7. Освен че забелязва заявки в orders.txt, **Volt забелязва** и **готова продукция в production.txt,** при което я **премахва от там**, както и **премахва** съответната клиентска **заявка от orders.txt,** след което **записва** успешно данните за заявката **в history.txt.**



Project Metrics

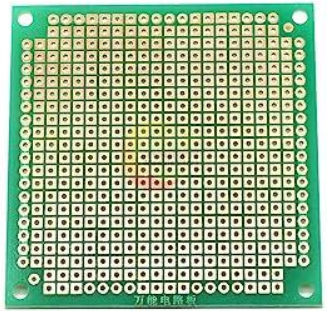
Не работим в конкретни мерни единици (нанометри, милиметри..).

Репрезентираме физическите характеристики (размери, позиция на pins) на компонентите в дискретен 2d grid в равнината:

2N3904

7x3

```
0000000
0102030
0000000
```



Project Terminology

Electronic Component

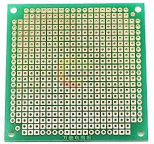
Съставна част, от която се създават хардуерни компоненти.

- Всеки компонент има 1 или повече входно/изходни връзки.
  - Ще наричаме такава връзка: **pin**
- Всеки компонент има **n x m** размери, където n и m могат да варират.
- Един компонент може да бъде завъртян по 4 различни начина при разполагане върху хардуерен компонент (на 90 градуса)
- Компонентите са ресурс, който трябва да се поръчва и от който се изработва финалния продукт.

Board

Съставна част, със свързваща функция, върху която се поставят електронни компоненти.

- Компонентите се на определени позиции с определени ротации (според дизайна на хардуерния компонент).
- Съществуват само в n x n размер, където n може да е: 8, 16, 24, 32, 64, 128, 256 и 512.
- На нея се принтират и връзките между pin-овете на отделните електронни компоненти.



Hardware Component

Финалния продукт, състоящ се от:

- основа, направена от **board**,
- разположени на нея електронни компоненти и
- множество от проводникови връзки между pin-овете на различните електронни компоненти.

Може да е с n x m размери, където n и m варират.

CODE ACADEMY

Project Terminology

Client Order

Клиентска поръчка. Има следните характеристики:

- Уникален целочислен идентификатор на поръчката (който се записва и показва в hex формат), генериран от Ohm при направа на поръчката.
- Име на клиент.
- Приоритет (normal, high)
- Дизайн на харуерен компонент (нужен само ако е за нов продукт, за който системата няма информация)
- Дизайни на електронни компоненти, използвани в продукта (нужни само за непознати на системата електронни компоненти).
- Поръчките се приемат и записват в реда, в който са направени в orders.txt. Обработват се по приоритет:
  - Започнатите поръчки са първи.
  - Следват поръчки, маркирани като high priority
  - На последно място се обработват останалите поръчки, в реда, в който са направени.

Printer Job

Заявка за серийно принтиране на хардуерен компонент в определено количество от определен принтер.

- Специфицира хардуерния компонент, който следва да бъде принтиран.
- Специфицира желаното количество за продукция (брой еднакви хардуерни компоненти).
- Извършва се на слоеве и отнема време, което симулираме със системни Sleep функции.

CODE ACADEMY

Component Libraries

Системата поддържа “база данни” (текстови файлове) с **дизайните на всички познати електронни и хардуерни компоненти**.

Когато клиент направи поръчка за несрещан хардуерен компонент, който може да има несрещани електронни компоненти, **дизайните на новите компоненти** (подадени от клиента, ако поръчката е валидна) **се записват в Electronic Components Library и Hardware Componentes Library**.

Всяка следваща поръчка на същия компонент няма нужда да съдържа дизайн на компоненти, при положение, че се очаква, че информацията за тях ще е вече налична - поръчката се брои за валидна и без необходимите дизайни.

Различните приложения на системата трябва да имат достъп до библиотеките за това, което им е необходимо.

CODE ACADEMY

Inventory

Инвентара съдържа складовата наличност на **ресурсите**, необходими за създаването на хардуерни компоненти (**видове** и **брой**):

- **Електронни компоненти**;
- **Boards**, които идват в различни фиксирани **n x n** размери, където n може да е **8, 16, 24, 32, 64, 128, 256 или 512**; За всеки тип board пазим налично количество.

Amp има **конфигурация за минимални количества**, които трябва да се поддържат за всеки ресурс и прави **автоматични поръчки** когато засече, че даден ресурс е под специфицираното количество.

Нови компоненти могат да се поръчват (и се пази наличното им количество) след като сме получили дизайн за тях при клиентска поръчка.

CODE ACADEMY

Component Formats

Има три различни начина (формата) за представяне както на електронен компонент, така и на хардуерен компонент:

- **Description** Format: текстов формат, описващ характеристиките на компонентите. Клиентските поръчки се правят в такъв формат (новите дизайни на компоненти).
- “**Machine-Level**” Format: текстов формат, “удобен” за четене от машина.
- **Visual** Format: текстов формат, подходящ за визуална репрезентация, използван за тестване, дебъгване и user interface.

- Библиотеките с дизайни на компоненти съдържат само първите два формата.
- Клиентски заявки се правят в Description Format.
- Принтерите познават и работят само с Machine-Level формат.
- Финалния продукт включва само първите два формата.
- Стъпките от процеса на производство могат да се принтират в които и да е от трите формата.

CODE ACADEMY

ASAPML Ω (Ohm)

Програма, която дава възможност за правене на поръчки от стандартния входен поток - от конзолата или от файл, ако използваме file redirection.

При успешно направена поръчка, информацията се добавя във файл, който съдържа текущите необработени поръчки.

Програмата трябва да дава възможност за:

- Добавяне на нова поръчка в orders.txt.
- Показване на поръчки, в реда, в който са направени.
- Показване на поръчки, филтрирани по приоритет.
- Отказване на поръчка.

Параметри, които приема по командния ред:

- релативния път от data директорията до cfg_ohm.txt, конкатениран с името на файла.
- релативния път от data директорията до orders.txt, конкатениран с името на файла.

CODE ACADEMY

ASAPML Amp

Програма, която дава възможност за автоматизирано следене на складовата наличност (в inventory.txt) и правене на поръчки на ресурси.

В конфигурационния файл на програмата има настройки за минималните количества, които трябва да се поддържат налични.

Програмата трябва да дава възможност за:

- Извеждане на текущия инвентар на екрана.
- Проверка за достатъчна наличност в инвентара.
- Поръчка на ресурси (след което обновява inventory.txt с новите количества).
- Настройка на минималните количества от страна на потребителя.

Параметри, които приема по командния ред:

- релативния път от data директорията до cfg_amp.txt, конкатениран с името на файла.
- релативния път от data директорията до inventory.txt, конкатениран с името на файла.

CODE ACADEMY

ASAPML Volt

Програма, която следи за:

- Незапочнати поръчки в orders.txt
- Готова продукция в production.txt
- Складова наличност на ресурси преди да започне поръчка в inventory.txt

Програмата също така:

- Записва успешни процеси на производство в history.txt
- Създава нови заявки за принтиране в print_jobs.txt.
- Обновява inventory.txt с използваните ресурси при добавяне на print job.
- Обновява библиотеките за компоненти с нови дизайни от клиенти.

Програмата трябва да дава възможност за:

- Принтиране на текущите print_jobs
- Изпълняване на следващите стъпки от процеса на производство (на базата на състоянието на файловете, описващи поръчки, инвентар и готова продукция)

Програмата приема релативни пътища до файловете, с които работи по командния ред.

CODE ACADEMY

ASAPML Bit

Програма, която чете за текущи заявки за принтиране в print_jobs.txt и ги изпълнява една по една, всяка - стъпка по стъпка.

Програмата симулира процес на принтиране по следния начин:

- Приема заявка, която специфицира модела на компонента за принтиране и бройката.
- Обръща се към библиотеката за необходимите дизайни на компонентите.
- Принтира финалния продукт на словеве (**Print Layers**) - отделни стъпки на принтиране:
  - Дъската на която се принтира се изрязва от оригиналната си n x n форма във формата на хардуерния компонент.
  - Започва се принтиране на проводниковите връзки между всички двойки свързани pin-ове на електронни компоненти.
  - Всеки вид компонент се принтира на отделна стъпка. (например: първо принтираме всички 2n3904 транзистори, след това принтираме всички ne555 чипове.

За всяка стъпка от принтирането използваме **матрица-маска** - списък от цели, положителни числа, които в бинарен формат специфицират позициите на клетките на принтирания компонент. Там където има 1 - принтираме клетка от компонент, иначе - не принраме нищо.

- Резултатите от всяка стъпка, заедно с използваната матрица за принтирането се извеждат на стандартния изход.
- Програмата работи и принтира на всяка стъпка само в Machine-Level формат.
- Само финалния резултат трябва да бъде принтиран и в трите формата.
- Всяка стъпка отнема определено време, което симулираме със Sleep системна функция и което се конфигурира в cfg_bit.txt
- При започване на процес на принтиране, заявката се премахва от print_jobs.txt. След приключване на принтирането, програмата добавя резултата в production.txt

В конфигурационния файл на програмата има настройки за времето, необходимо на всяка стъпка.
Програмата приема релативни пътища до файловете, с които работи по командния ред.

CODE ACADEMY

Electronic Component Design Formats

- **Description Format:**
  ```
  id: 2N3904
  width: 7
  height: 3
  pins: (1,1),(3,1),(5,1)
  ```

- **Machine-Level Format:**
  ```
  2N3904: 775 257 263 261
  ```

- **Visual Format:**
  ```
  2N3904

  0000000
  0102030
  0000000

  #comments in all formats
  ```

Machine-Level Dimensions (width, height and pin coordinates)

- **w x h = 7 x 3**
  ```
  7 = 00000111
  3 = 00000011
  ```

  8 bits for each number, 16 bits for both numbers (left is least significant):

  ```
  00000011 00000111 =
  ```
  775

- **(x,y) = (5,1)**
  ```
  5 = 00000101
  1 = 00000001

  00000001 00000101 =
  ```
  261

Hardware Component Formats

- **Description Format:**

```
id: Darlington_Pair_0
width: 11
height: 7
components:
2N3904: (0,0),0                                    # (x,y),rotation
2N3904: (10,0),1
connections: (1.3,2.2), (1.1,2.1)
```

- **Machine-Level Format:**

```
Darlington_Pair_0: 775 257 263 261
      2N3904: 2 0 0 10 1                           # TOTAL_COUNT 2N3904 (x,y) r 2N3904 (x,y) r
                                                              # in board coordinates: (5,1)-
1281 777 257 265
(9,3) (1,1)-(9,1)
```

- **Visual Format:**

```
Darlington_Pair_0

0000000*000
0102030*010
0000000*000
********020
********000
********030
********000
```

CODE ACADEMY

Допълнителни материали

https://www.youtube.com/watch?v=g8Qav3vIv9s&list=LL&index=2&t=136s&ab_channel=InterestingEngineering

https://www.youtube.com/watch?v=YJr-kHy6STg&ab_channel=Lesics

https://www.youtube.com/watch?v=iSVHp6CAyQ8&list=LL&index=1&t=594s&ab_channel=CNBC