AIP BASED DUTY SCHEDULING

Submitted in partial fulfillment of the course

CS-122 Computer Architecture and Organization (Jan – May 2022)

Author

-Bhavirisetty Venkata Harshit
(BT20HCS177)

Executive Summary

Data applications and handling are transforming the era of computer architecture and organization. There are so many emerging technologies in the field of memory, storage, computer and network. The trends revolving around storages like SMR and low latency NAND flash are bringing new changes and smooth mechanisms. Development of mobile processors in smartphone architecture plays an important role. They are mainly subject to the design metrics, cost and efficiency. The processor architecture should support more complex user interface and dynamic design which is also the requirement in the development of our application. To provide these advanced features we require complex hardware coprocessors.

In this paper we are going to focus on the features and elements required for the development of database and scheduling mobile applications and the structure of mobile processors supporting our application and objective.

Introduction of new upcoming technologies

There are distinctively two approaches for the implementation of cellular handsets which includes programmable DSP's while the other approach utilizes Application Specific integrated circuit techniques. DSP architecture is preferred over ASIC because of the global system communication. Conventionally, DSP used Harvard architecture which is in contrast with the

Von Neumann Architecture. In Harvard architecture it physically separates storage and pathways for instruction and data whereas in Von Neumann data and instructions are in the same memory. New DSP architecture like TMS320C55 implements harvard architecture utilizing less read buses for code and data which gives programmable idle modes with automatic power saving options. Due to advancement in chip fabrication there is a great improvement in computational power in modern DSP.

Artificial intelligence was a concept way before the development of microprocessors but with advancements in both theories the intermission led to development of AI in computer architecture. AI has supported different tasks like SIMD, TPU and GPU with efficient algorithms for better computer architecture and performance. The first practical implementation of AI was a digi recognizer called LeNet. This led to development of different hardwares and AI started to flourish under machine learning. There was development of AI models like speech recognition by microsoft based in deep neural networks in 2011. ImageNet was also successful in decreasing the error rate by utilizing convolution neural networks. However these AI models required more space and frequency for the working which leads to thread level parallelism using multiple processors which boosted the performance to next level producing real time processing of AI.

Smartphones now contain system on chip (SoC) devices that provide data processing and interfacing with other ongoing systems in mobile devices. This helps in transfer of data from sensors and connectivity to transport.

As the 5G has already set its vision high up, the 6G is coming up next with next level change in performance in the response time. It will also enable critical application of real time automation and sensing for the physical world. 6G is designed as more power efficient and sustainable for a greener economy.

Mobile Application Architecture

Mobile application architecture is an outline for structure, process and patterns to develop a successful mobile application with all the requirements and standards for user and developer. Technical frameworks make up the front and back end of a mobile app which is judged upon different questions like the cost, maintenance status, test and error. A good mobile application architecture will depend on the software development principles like KISS, DRY and SOLID in different stages of development of mobile application. The data flow should be maintained in a way which supports a clear data flow for longer run or expansion in the industry.

There are three layers of mobile application architecture :

1. **User Interface (UI) and User Experience (UX)**
   This layer is concerned with the interaction between user and the application. The objective is to make the interface more user friendly and efficient for use. UI is concerned upon the design concepts and ux for the customer interaction. The first step to achieve in UI and UX is research to make sure the compatibility of the app. This is followed by the

tests and discussion for the behavior and response on the data. The interface is then released accordingly.

2. **Business Layer**
   This layer is about the data exchange and security of the application. It is responsible for loading of the data from different networks and databases. And processing the data for presentation purposes. It contains two layers which are domain model and service layer. Domain model contains information more user related. This layer looks upon the security, data caching, logging and data validation. It should point out the complexities related to the problem space. Service layer is more of atomic work done on domain models. It also encapsulates the business logic and controls transaction responses.

3. **Data Layer**
   The data layer contains data utilities, service agents and access to support transactions. The layer contains two parts which are persistence and network. Persistence layer works as an intermediate between the business function of the app and the data storage. It gives data access with sources via API. Network helps in the communication for the connection of transfer of data packets between different networks. It also helps in routing and error reporting.

The most commonly acceptable architecture for android development is Clean Architecture. It is built upon the concern related to the platform specific and platform independent functionalities. The main feature of clean architecture is that UI and data source can be easily changed.
There are five principles which are applied in software development with clean architecture (SOLID) :

1. **Single Responsibility Pattern**
   A module should be responsible for the one action that can change actions affecting by another action also looking under the deployment.

2. **Open Close Principle**
   Extension required for modification should not be a massive change in code.'

3. **Liskov Substitution principle**
   Everything associated should have a contract with the interface so that it is easy to replace by the client for its derivative class.

4. **Interface Segregation Principle**
   We should create multiple interfaces as the functionality is used by several clients.

5. **Dependency Inversion Principle**
   The project's dependency should be on abstractions like interfaces and abstract classes rather than concretions as to allow injecting dependency for purposes.

Hybrid mobile architecture is both native and web solutions. It uses shells at the backend but uses Javascript, HTML and CSS for the front-end. It uses plugins such as Apache Cordova to access native platform features. Hybrid mobile apps are easy to upgrade but not fit for complex and interactive features. Single Codebase for multiple platforms and very cost efficient. Technologies and frameworks used which are react antive, Xamarin and ionic.

Cross platform application architecture works on a common codebase with platform specific abilities in each native shell. It relies on frameworks rather than web language. It has a separate and distributed codebase for each platform. The system efficiency and attractiveness make it easier to switch higher investments.

Packaging of your android application requires creation of an APK file and for iOS requires an IPA file. Both of the files can be generated in Cordova for the deployment in your mobile phone.

As a part of the full lifecycle application development service set, companies have been offering custom software development services to clients belonging to a vast range of industries for a very long time. Some advice of designers and developers generally share when it comes to choosing the best mobile app architecture diagram are:-

* To build native software which provides intuitive performance and functionality.
* Wide user base with cross-platform application development.
* To engage your customers and your internal stakeholders, the creation of web and native software development to ensure business visibility and give the customers the option to access the offering on multiple devices.

Introduction to AIP scheduling

AIP or Admission Interaction Process is a unique procedure undertaken by the university which involves aptitude tests and interaction with the student via video conferencing or face-to-face. Therefore, to evaluate the student and have a one-to-one conversation with their parents, various faculty members are being allotted accordingly. The process of allotting faculty members for AIP is entirely manual as it involves scrolling through the time- table of the faculty members, looking at the free slot and assigning the duty accordingly. Therefore, we have tried to automate this process to some extent by providing two interfaces- the admin side and the teacher side. Hence, together with these two interfaces, we have tried to build an application and expect to extend its functionality in the upcoming semesters as well.

Mobile Application Features

Current features-

- ☐ Two interfaces- Admin and Teacher
- ☐ View time- table of the faculty
- ☐ Admin can assign the duty according to the available free slots

- ☐ Admin may make a change (update, delete or view) if required
- ☐ Teacher interface- shows allotted slots, numbers of hours dedicated to AIP and an option to either accept or decline the duty according to their schedule.

For future-

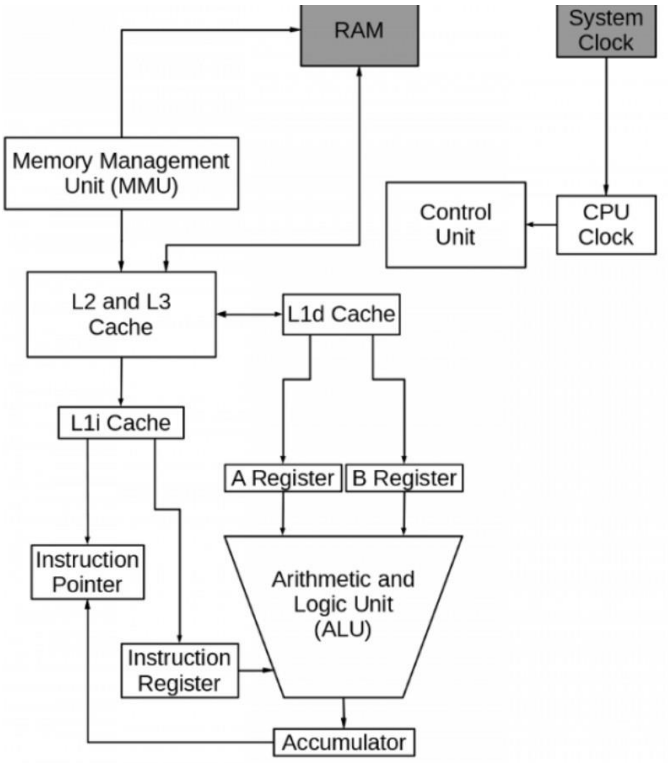
- ☐ Addressing the security issues related to cloud connectivity
- ☐ View upcoming AIP's with respect to a new student logging in everytime
- ☐ Introduction to fingerprint/ password in the application itself
- ☐ Automatic capturing of the free slots and assigning the duties to the set of faculties provided.
- ☐ Automating the procedure of calculating honorarium based on the position and number of hours dedicated to AIP by the faculty member.

Mobile Processor Architecture-

The smartphone hardware primarily consists of application processors (SoC), RAM (mobile SDRAM/ mobile DDR), DSP, CPU (ARM processors) etc. Processors that are used in mobile phones concentrate more on cost, current market requirements and use of low power. Since power, cost and real- time computation requirements are the constrained resources, therefore different processors have distinct characteristics and a limited programmability.

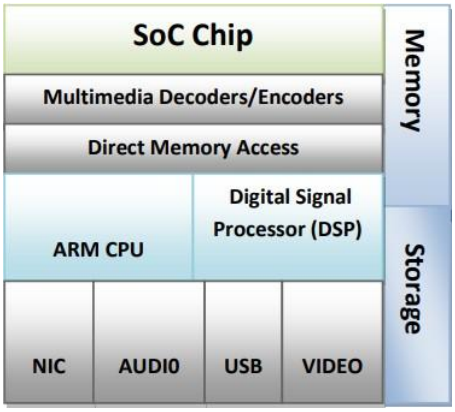
However, it is important that the processor architecture of the mobile phone delivering data services must be capable of bearing much complex user interface, dynamic operating environments and additional services. Therefore, one may require multiple DSP's or hardware coprocessors.

There are two major approaches for the implementation of the mobile phones; one being the programmable DSP's and the other ASIC or the Application- specific integrated circuit techniques. DSP are the specialized microprocessors which were designed around single stand-alone integrated circuits. These are still dominant in the wireless handset market for digital cellular telephony as they provide real- time operation at low power costs. DSP is based on Harvard Architecture where there are separate pathways for storage and signal processing instructions and data. This is opposite of the Von Neumann Architecture or the Princeton Architecture where both data and instructions are saved in the same memory. Due to simultaneous transmission of data in Harvard Architecture, it results in fewer cycles for executing a function and enables high memory bandwidth with multiple operand functions.



System on Chip (SoC) based Architectures-

This is a type of enhanced DSP hybrid chip which can manage real time responsiveness. Lowering the voltage of the chip enables low power consumption. In SoC, system tasks can be managed by integrating microcontrollers, dedicated ASIC's or DSP's in a single chip. Higher throughput of VLIW architecture requires faster memory systems (like cache memories). Various companies customize the instruction sets of these processors to speeden up the read- write operations. The most used instruction set is the 16- bit.



Harvard Architecture

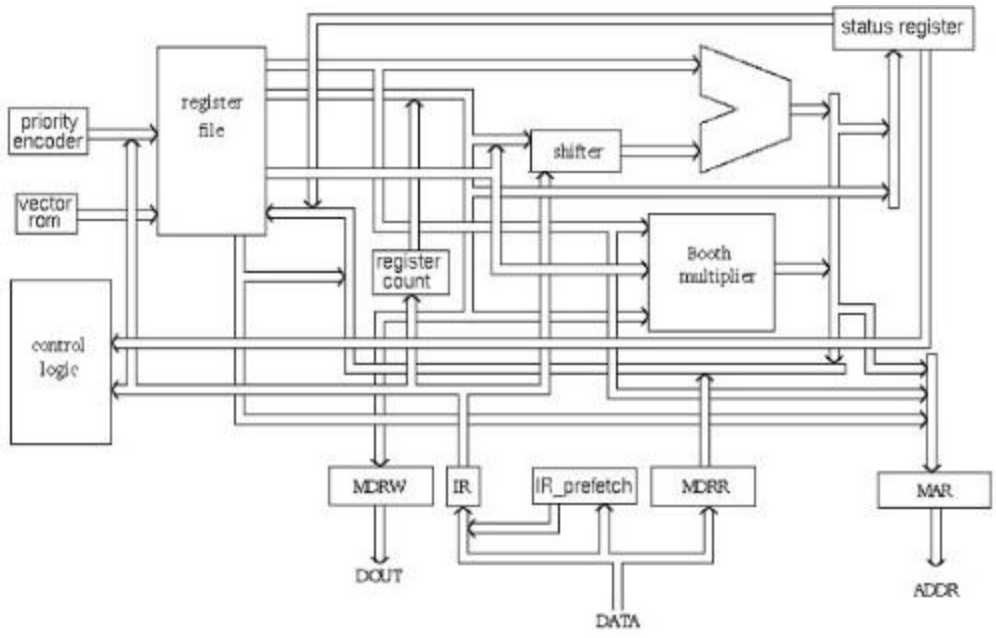
The ARM Architecture consists of

- It has two types of physical address spaces- tag physical address space, and data physical address space. The Tag load and store instructions are used to access Allocation tags in a tag physical address space. Whereas the data physical address space is accessed by data load and store instructions to access data.
- The Allocation Tags are of 4- bit, where one Allocation Tag is mapped to a set of naturally-aligned 16 physically addressed locations.
- It implements page level control over the access made to the Allocation Tags in memory. It uses System register for this.
- It generates a Logical Address Tag from which a Physical Address Tag can be derived. This is then contained in the upper bits of the address that is used in load and store data instructions.
- A Tag Check of the Physical Address Tag is also performed against the 4-bit Allocation Tag for each data location that is accessed by a Tag Checked load or store of data.
- It also has Instructions to generate and manipulate the Logical Address Tag in a register.

In today's world, we emphasize more on single or dual- core CPUs which provide a better overall performance.

ARM Based Processors

The ARM architecture is the most pervasive group of reduced instruction set computing (RISC) architectures used for computer processors. It provides a high performance with energy efficiency, integrated security and a large ecosystem for global support. The Arm CPU architecture is implemented with numerous other microarchitectures to provide software compatibility. The CPU architecture defines the basic instruction set, and the exception and memory models that are further based on the OS. It determines how implementation meets architectural contract by defining the design of the processor and covering important aspects such as pipeline length, area and levels of cache.



ARM Block Diagram

The ARM architecture is well known for its variety of cores in the market, letting the designers choose according to their applications ranging from low power cores in the automation market to high- performance cores in the Health and avionic sectors. There are three main categories of this cortex family namely

1) Cortex-A (Application Processor Cores)
2) Cortex-R (Real Time Application cores)
3) Cortex-M (Microcontroller Cores)

We have used Cortex-A processors which are widely known for their application related to the Android and Linux Operating Systems. Our comparison will be based on the latest Cortex-A77 and A78 released processors and its future compatibility with our developed application.

Instruction Set: ARMv8.5-A: Advanced RISC Machines

It is a family of reduced instruction set computer (RISC) instruction set architectures for computer processors.

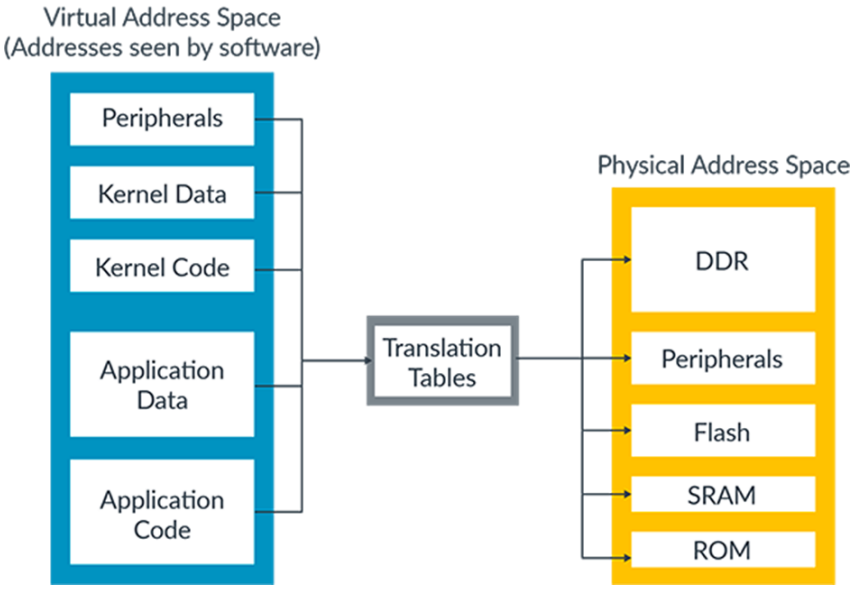
- ARMv8-A was released in the year 2011, and this architecture added support for a 64-bit address space and 64-bit arithmetic.
- Its instruction set has a fixed length of 32- bit. It includes the following RISC features:
  - ➔ Its architecture is a basic Load/store architecture.

- ➔ The array of processor registers is a uniform 16 × 32-bit which includes the registers like program counter, stack pointer and the link register.
- ➔ The instruction is of fixed width, ie, 32 bits. It eases instruction decoding and pipelining, at the cost of decreased code density. Later, the Thumb instruction set added 16-bit instructions also and hence, increased code density.
- ➔ Most of the instructions here require a single clock-cycle to execute.

Memory Management

Memory management describes how access to memory in a system is controlled. The address received by the software executing on a processor is virtual addresses. This virtual address is translated into a physical address by the processor. These physical addresses point to the actual physical locations of the memory and are therefore used for accessing the memory.

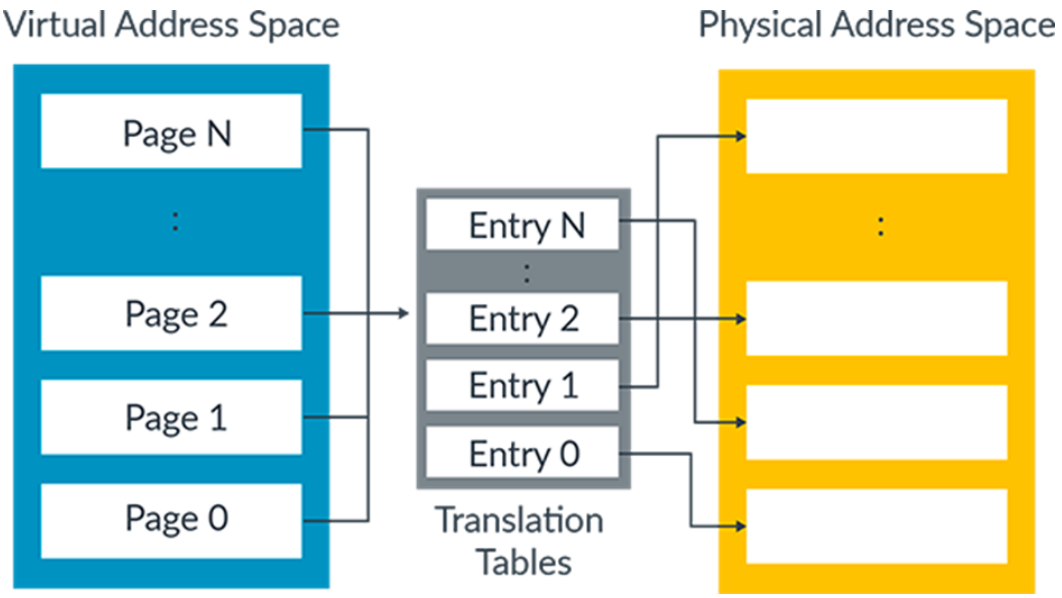
Virtual addresses are translated to physical addresses through mappings. The mappings between virtual addresses and physical addresses are stored in translation tables (or page tables). The following diagram shows this:



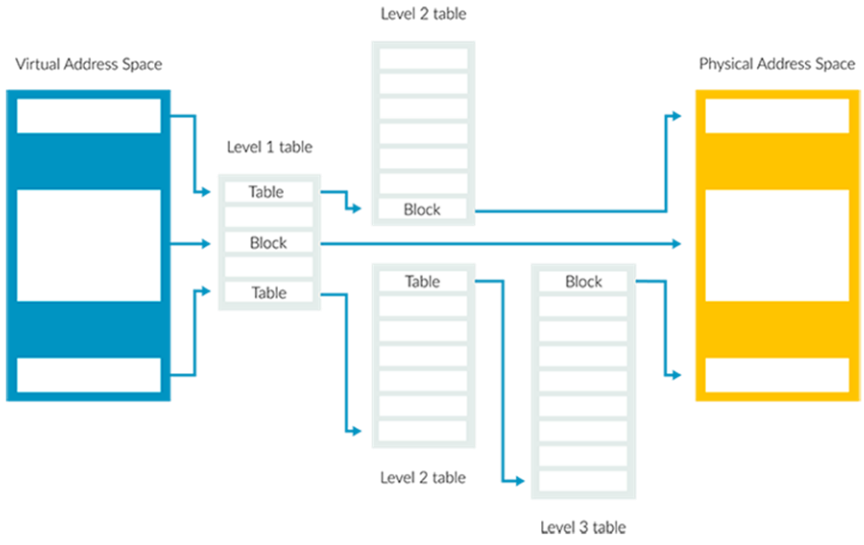
Translation tables are stored in memory. These are managed by the OS.

The translation tables work by dividing the virtual address space into equal-sized blocks and by providing one entry in the table per block. Each entry contains the address of a corresponding

block of physical memory. This is used while accessing the memory through a physical address. This is shown as follows:



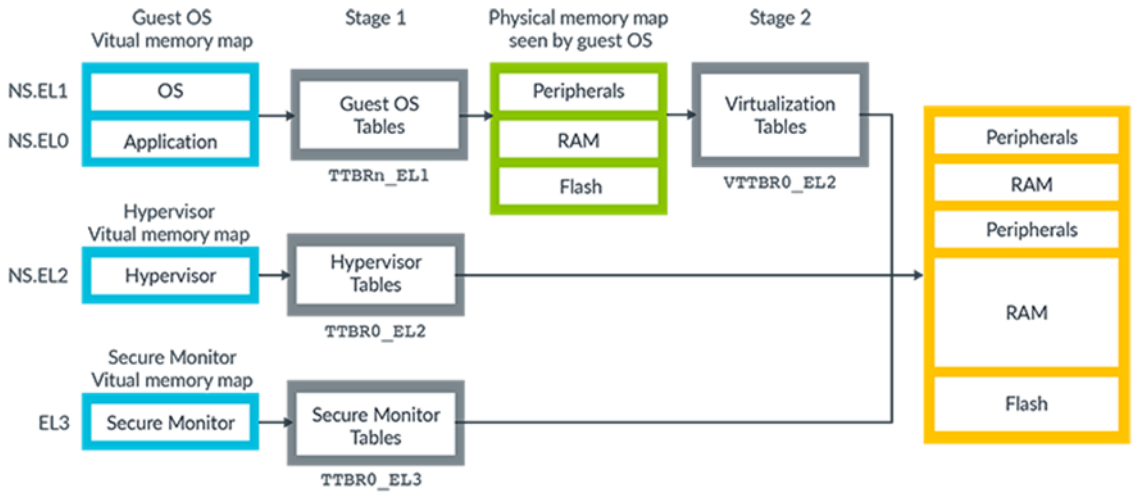
In a single-level lookup, the virtual address space (VAS) is split into equal-sized blocks. But in practice, a hierarchy of tables is used. Hence, there comes multi- level lookup.
An example of a multilevel table that has three levels is as follows:



The Level 1 table divides the VAS into large blocks. Each entry in this table can point to an equal-sized block of physical memory or it can point to another table. That table too subdivides the block into smaller blocks. We call this type of table a 'multilevel table'.

In Armv8-A, the maximum number of levels is four, and the levels are numbered 0 to 3. This multilevel approach allows both larger blocks and smaller blocks to be described.

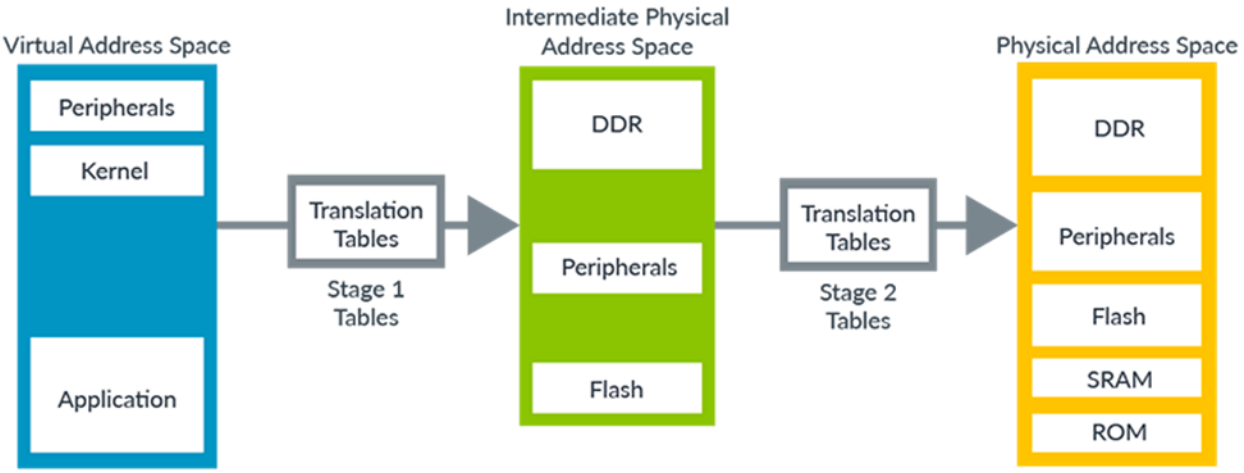
There are several independent virtual address spaces in Armv8-A. This diagram shows these virtual address spaces:



The diagram shows three virtual address spaces:

- NS.EL0 and NS.EL1 (Non-secure EL0/EL1)
- NS.EL2 (Non-secure EL2)
- EL3

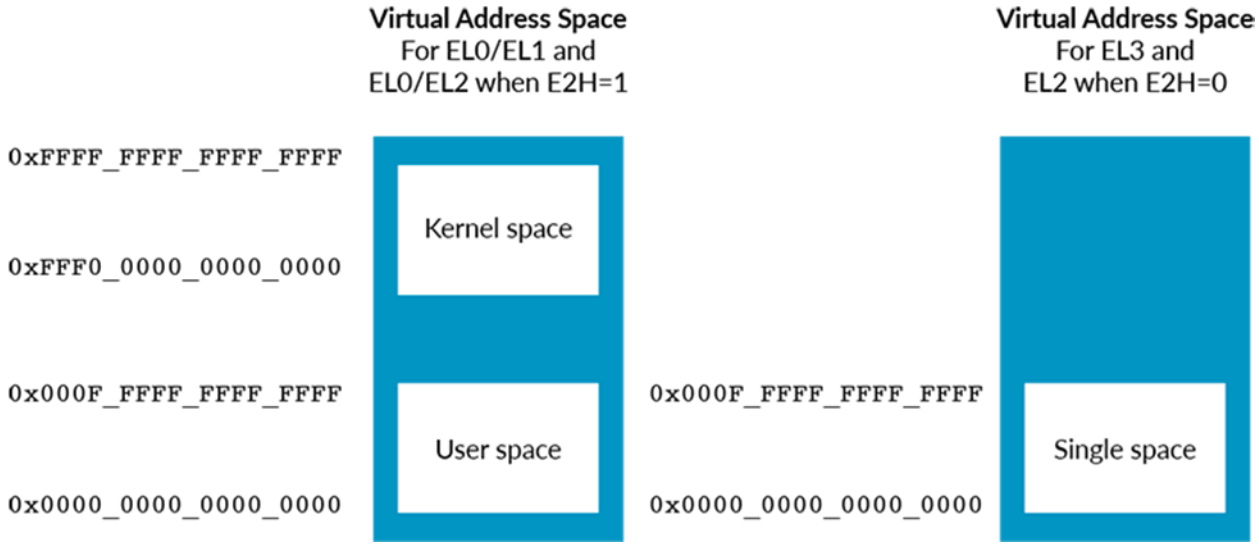
Support for Secure EL2 was added in Armv8.4-A. The translation of virtual address space to physical address space happens as follows:



Armv8-A is a 64-bit architecture, but all addresses in it are not 64-bit.

Size of Virtual addresses:
Virtual addresses are stored in a 64-bit format. As a result, the address in load instructions and store instructions is always specified in a register. However, not all of the addresses in that register are valid.

This diagram shows the layout of the virtual address space in AArch64:



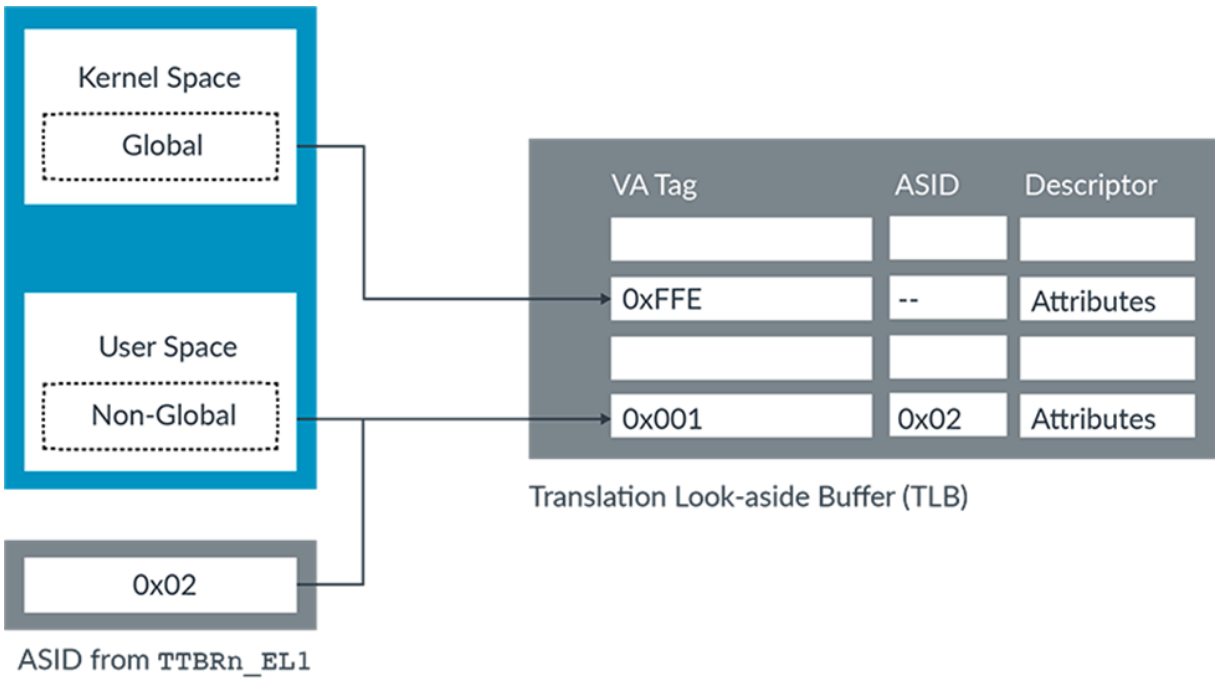
All Armv8-A implementations support 48-bit virtual addresses.

Size of Physical Addresses:
The size of a physical address is IMPLEMENTATION DEFINED, up to a maximum of 52 bits. For Arm Cortex-A processors, this is usually 40 or 44 bits.
In Armv8.0-A, the maximum size for a physical address is 48 bits. This was extended to 52 bits in Armv8.2-A.

Address Space Identifiers:
Armv8-A processors use Address Space Identifiers (ASIDs) for knowing which version of translation to use.

For the EL0/EL1 virtual address space, translations can be marked as Global (G) or Non-Global (nG) using the nG bit in the attributes field of the translation table entry. Global translations apply whichever application is currently running. Non-Global translations only apply with a specific application.

Non-Global mappings are tagged with an ASID in the TLBs. On a TLB lookup, the ASID in the TLB entry is compared with the currently selected ASID. If they do not match, then the TLB entry is not used. This diagram shows a Global mapping in the kernel space with no ASID tag and a non-Global mapping in user space with an ASID tag:



Transition Granule:
A translation granule is the smallest block of memory that can be described. Larger blocks are the multiples of granule.
Armv8-A supports three different granule sizes: 4KB, 16KB, and 64KB.

The granule sizes that a processor supports are IMPLEMENTATION DEFINED. All Arm Cortex-A processors support 4KB and 64KB. This table shows the different block sizes for each level of table based on the selected granule:

Level of table	4KB granule		16KB granule		64KB granule	
	Size per entry	Bits used to index	Size per entry	Bits used to index	Size per entry	Bits used to index
0	512GB	47:39*	128TB	47*	-	-
1	1GB	38:30	64GB	46:36	4TB	51:42
2	2MB	29:21	32MB	35:25	512MB	41:29
3	4KB	20:12	16KB	24:14	64KB	28:16

There are restrictions on using 52-bit addresses. When the selected granule is 4KB or 16KB, the maximum virtual address region size is 48 bits.
Similarly, output physical addresses are limited to 48 bits. It is only when the 64KB granule is used that the full 52 bits can be used.

Study and Comparison of Mobile Processor Architecture

The three processors on which we are going to experiment our application are:

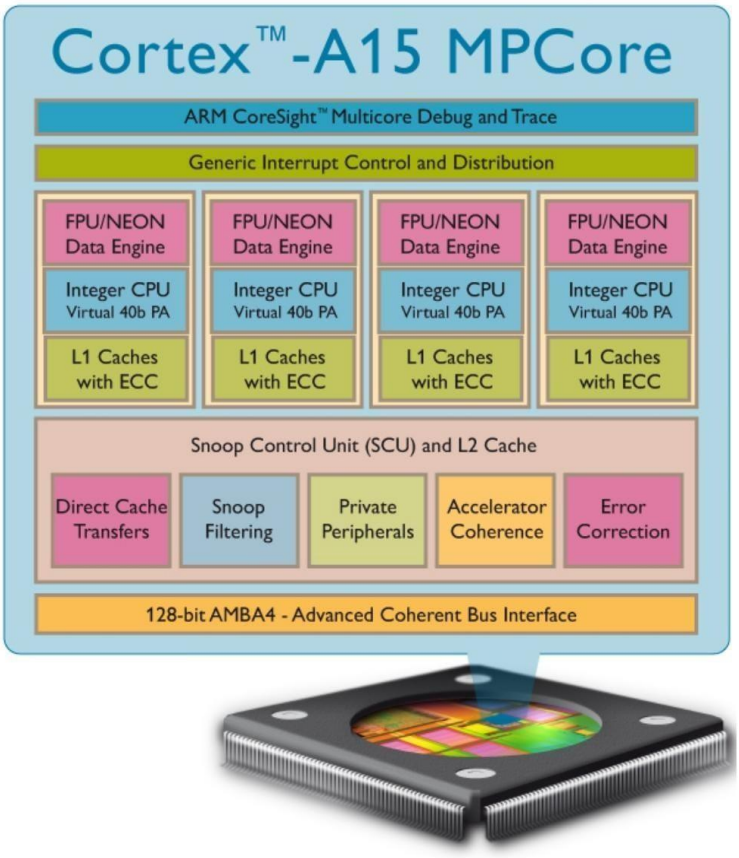
1) Apple Bionic A15 Chip (by Apple)
2) Samsung Exynos 2100 5G (by Samsung)
3) Qualcomm Snapdragon 865+ (by Qualcomm)

1) Apple Bionic A15 Chip

Officially announced on September 14, 2021, this chipset is made using a 5-nanometer process technology. It has 6 cores out of which 2 cores are Avalanche at 3223 MHz while the remaining 4 cores are Blizzard at 1820 MHz.

This chip is based on the ARMv8 Instruction set and has a 16-core neural engine. Various other features of this chip are mentioned below

- ➔ Contains 15 billion transistors
- ➔ Provides 64-bit support
- ➔ L1 Cache- 192 KB
- ➔ L2 Cache- 8MB
- ➔ Thermal Design Power at 6W
- ➔ GPU- Apple A15 GPU
- ➔ Memory Type- LPDDR4X
- ➔ Memory Frequency at 4266 MHz
- ➔ Maximum Bandwidth- 42.7 Gbit/s
- ➔ Max size- 8 GB
- ➔ 4G Support- LTE Cat. 24
- ➔ Provides 5G support with 6 Wifi(s)
- ➔ Can be seen in flagship class phones with model number APL1W07 (iPhones, iPad, etc)



Block Diagram of A15 Bionic Chip

2) Samsung Exynos 2100 5G

Manufactured in 5LPE at Samsung, this chipset was officially announced on December 2, 2020 using 5 nanometer process technology. With the codename of Exynos X1/ Cortex A- 78/A-55, this chip is based on ARMv8.4-A instruction set.

- ➔ Three major architectures- SoC with 8 cores in 3 clusters (1x Exynos X1 2.9 GHz/ 3x Cortex-A78 2.8 GHz/ 4x A55 2.2 GHz)
- ➔ Frequency at 2900 MHz
- ➔ Integrated ARM Mali-G78MP14 GPU (+40% faster versus Mali-G77MP11)
- ➔ L1, L2 and L3 cache size, transistor count not revealed yet
- ➔ Thermal Design Power at 9 W
- ➔ Memory type- LPDDR5
- ➔ Memory frequency at 3200 MHz
- ➔ Uses 4x16 Bit bus
- ➔ Maximum Bandwidth at 51.2 Gbit/s
- ➔ Maximum Size- 16 GB
- ➔ Provides 4G support- LTE Cat. 25
- ➔ Has 5G support as well with a download speed of 3000 Mbps and upload speed of 422 Mbps, 6 Wifi(s)
- ➔ Image Compression Rate at 170.5 MP/ second, face detection rate at 26.7 images/ second
- ➔ Speech recognition rate at 64.15 words/ second
- ➔ HTML5 rate at 3.21 Mnodes/second
- ➔ SQLite rate of searches at 970.15 thousand rows/ second
- ➔ Found in flagship phones with model number S5E9840

3) Qualcomm Snapdragon 865+

Popularly known for its spectacular performance and superior gameplay, this chipset is a truly global 5G and ultra- intuitive AI with an unstoppable platform. It was announced in July 2020 and is found in flagship android phones with model number SM8250- AB.

- ➔ 8 cores; [1x 3.1 GHz – Kryo 585 Prime (Cortex-A77), 3x 2.42 GHz – Kryo 585 Gold (Cortex-A77), 4x 1.8 GHz – Kryo 585 Silver (Cortex-A55)]
- ➔ Frequency- 3100 MHz
- ➔ Used Instruction set- ARMv8.2-A
- ➔ L1 Cache size- 512 KB
- ➔ L2 Cache Size- 1 MB
- ➔ L3 Cache Size- 4 MB
- ➔ Uses 7 nm process technology

- ➔ Thermal Design Power at 10 W
- ➔ GPU- Qualcomm Adreno 650
- ➔ Memory type- LPDDR5
- ➔ Memory frequency at 2750 MHz
- ➔ Uses 4x16 Bit bus
- ➔ Maximum Bandwidth at 44 GBit/second
- ➔ Maximum size - 16 GB
- ➔ Modem- X55
- ➔ Provides 4G support- LTE Cat. 24
- ➔ Provides 5G support; with download speed of 2500 Mbps and upload speed of 316 Mbps; 6 Wifi(s)

On comparing these processors together, it was found that Apple A15 Bionic > Samsung Exynos 2100 5G > Qualcomm Snapdragon 865+

Mobile Processor and AIP App

App size ~ 8-10 MB

The application can be uploaded on the playstore once the cloud connectivity is fully established. There might be an effect on the processing speed of the application while we switch on to different processors. However, there is no significant change observed in the working of the application.

Conclusion and Future Enhancement

The project brings out so many learning outcomes as it is a mixture of dimensions of learning which enhances not only the presentation form but also the deep rooted process behind the working of a mobile application. Some of the future enhancements that we plan to do are

1) Customization:
   When it comes to organizing the applications, the users may have the choice to design their UI according to their wishes. The mobile app is having more customization features like a web app. Various personalization options that we plan on including are: changing font text, size, the color of the text in the app, background image, day/night mode of the app etc.

2) More automation:
   Of course, simplicity is the key to good performing applications. UI/UX design (User Interface/ User Experience) is a crucial factor that should be implemented in every application to draw the attention of audiences. But we also plan on advancing the AI in a way so that if the teacher sends a notification for him not being able to attend the allotted slot, The AI will automatically adjust the schedule without any manual assistance.

3) Different modes of work:
   Most of the mobile apps are offline and some of the apps are online. But both will have a particular traffic base to access them. Thus, we plan to provide the users to be able to change the app mode based on their network. Offline mode apps are highly appreciated as the users do not worry about the bad network. But if the online mode is necessary to include, then making sure all of the features are working properly.

4) Regular Updates:
   A mobile app is always having regular updates to enhance the old features with new features. New features will result in more use of application.

At the final stage of our idea, we've planned to give regular up-gradation to our application, once we've created and published it to our audience. This is the path to retain old visitors and to get new audiences.
The scope of improvement goes as the technology grows in this field and will definitely bring out something intelligent from the upcoming. The goal of this project is to make the University's admission process more effective and efficient.

References-

1) https://www.triveditech.com/types-of-processor-used-in-smartphones/
2) https://www.arm.com/
3) https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/arm-ethos-n78-npu
4) https://www.qualcomm.com/products/technology/processors/mobile-processors
5) https://nanoreview.net/en/soc-list/rating
6) https://link.springer.com/chapter/10.1007/978-3-030-93677-8_48
7) https://ieeexplore.ieee.org/document/9516652
8) https://www.netsolutions.com/insights/mobile-app-architecture-guide/#:~:text=A%20clearly%20defined%20mobile%20app,the%20short%20and%20long%20term.
9) https://magora-systems.com/mobile-app-development-architecture/

10) https://nix-united.com/blog/the-comprehensive-guide-to-mobile-application-architecture/

11) http://thinkapps.com/blog/post-launch/mobile-app-performance-tips/

12) https://www.peerbits.com/blog/all-about-app-architecture-for-efficient-mobile-app-development.html

13) https://www.snia.org/educational-library/emerging-trends-computer-architecture-2019

14) https://www.clock.co.uk/insight/how-to-build-test-share-and-publish-a-hybrid-mobile-app

15) https://www.qualcomm.com/products/application/smartphones/snapdragon-8-series-mobile-platforms/snapdragon-865-plus-5g-mobile-platform

16) https://www.arm.com/architecture/cpu

17) https://gsasindia.com/newsroom/cortex-a-cortex-r-and-cortex-m/

18) https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.677.26&rep=rep1&type=pdf

19) https://www.macs.hw.ac.uk/~hwloidl/Courses/F28HS/Docu/DEN0013D_cortex_a_series_PG.pdf

20) https://www.qualcomm.com/media/documents/files/qualcomm-snapdragon-865-5g-mobile-platform-product-brief.pdf

21) https://nanoreview.net/en/soc/apple-a15-bionic

22) https://www.sammobile.com/samsung/exynos/2100/

23) https://developer.arm.com/documentation/101433/0101/Functional-description/Introduction/About-the-core

24) Arm_cortex_x1_trm_101433_0101_05_en.pdf

25) https://nanoreview.net/en/soc/samsung-exynos-2100

26) https://nanoreview.net/en/soc/qualcomm-snapdragon-865-plus

27) https://www.engadget.com/2010-09-09-arm-reveals-eagle-core-as-cortex-a15-capable-of-quad-core-compu.html

28) https://www.educba.com/exynos-2100-vs-a14-bionic/

29) https://versus.com/en/apple-a15-bionic

30) https://en.wikipedia.org/wiki/LPDDR

31) https://www.redhat.com/sysadmin/cpu-components-functionality

32)