

**MSPM’S**

**Deogiri Institute of Engineering and Management Studies, Aurangabad**

**Department of Basic Science and Humanities**

Report on

Computer Architecher & organization

Submitted By

**1)Aniket ghate (26071)**

**2) Sudhanshu Brahmaraj (26072)**

Under the Guidance of

Prof. Pankaj Durole

Department of Basic Sciences and Humanities

(Deogiri Institute of Engineering and Management Studies)

2018-2019

****

CERTIFICATE

This is to Certify that Mr. **Aniket ghate** and Mr. **Sudhanshu Brahmaraj** Seat No.26071 & 26072 has completed a report writing on Computer Architecher & organization.

**Pankaj Durole**  **Dr.S.Kalyankar Dr.UlhasShiukar**

**Subject Teacher H.O.D. Director**

## SPECIFICATIONS

### Dimensions

162.6×75.9×8.8mm

### Weight

206g

### Back Material

3D Corning® Gorilla® Glass

### Colors

Mirror Gray/Nebula Blue/Almond

### Operating System

OxygenOS based on Android™ 9

### CPU

Qualcomm® Snapdragon™855 (Octa-core, 7nm, up to 2.84 GHz), with Qualcomm AI Engine

### GPU

Adreno 640

### RAM

6GB/8GB/12GB LPDDR4X

### Storage

128GB/256GB UFS 3.0 2-LANE

### Sensors

In-display Fingerprint Sensor, Accelerometer, Electronic Compass, Gyroscope, Ambient Light Sensor, Proximity Sensor, Sensor Core, Laser Sensor

1. Battery

4000 mAh (non-removable) Warp Charge 30 Fast Charging (5V/6A)

### Vibration

Haptic Vibration

### Buttons

Gestures and on-screen navigation support  
Alert Slider

### Audio

Dual stereo speakers  
Noise cancellation support  
Dolby Atmos®

### Unlock Options

In-display Fingerprint  
Face Unlock

## Display

Size 6.67 inches(The corners of the screen are within a standard rectangle. Measured diagonally, the screen size is 6.67 inches in the full rectangle and 6.46 inches accounting for the rounded corners.)

### Resolution

3120 x 1440 pixels 516ppi

### Aspect Ratio

19.5:9

### Type

Fluid AMOLED

### Cover Glass

3D Corning® Gorilla® Glass

## Rear camera

### Rear camera - Main

Sensor: Sony IMX586  
Megapixels: 48  
Pixel Size: 0.8 µm/48M; 1.6 µm (4 in 1)/12M  
Lens Quantity: 7P  
OIS: Yes  
EIS: Yes  
Aperture: f/1.6

### Telephoto Lens

Megapixels: 8  
Pixel Size: 1.0µm  
OIS: Yes  
Aperture: f/2.4

### Ultra Wide Angle Lens

Megapixels: 16  
Aperture: f/2.2  
Field of View: 117°

### Flash

Dual LED Flash

### Lossless Zoom

3×

### Autofocus

Multi Autofocus(PDAF+LAF+CAF)

### Video

4K video at 30/60 fps  
1080P video at 30/60 fps

Super Slow Motion: 1080p video at 240 fps, 720p video at 480 fpsTime-Lapse  
Video Editor

## Front Camera

### Front Camera

Sensor: Sony IMX471  
Megapixels: 16  
Pixel Size: 1.0 µm  
EIS: Yes  
Autofocus: Fixed Focus  
Aperture: f/2.0

### 51-QMzEE0HL._SX569_.jpg

### ONE PLUS 7 PRO

### Video

1080P video at 30fps  
Time-Lapse

### Features

Face Unlock, HDR, Screen Flash, Face Retouching

## Multimedia

### Audio Supported Formats

Playback: MP3, AAC, AAC+, WMA, AMR-NB, AMR-WB, WAV, FLAC, APE, OGG, MID, M4A, IMY, AC3, EAC3, EAC3-JOC, AC4  
Recording: WAV, AAC, AMR

### Video Supported Formats

Playback: MKV, MOV, MP4, H.265(HEVC), AVI, WMV, TS, 3GP, FLV, WEBM  
Recording: MP4

### Image Supported Formats

Playback: JPEG, PNG, BMP, GIF  
Output: JPEG, PNG

* INSTRUCTION SET:

The ARMv8-A instruction sets

The A64 instruction set is similar to the existing A32 instruction set. The instructions themselves are still 32 bits wide and have similar syntax. The instruction sets use a generic naming convention within the ARMv8-A architecture, so that the original 32-bit instruction set states are now called:

**A32:** When in AArch32 state, the instruction set is largely compatible with ARMv7-A, though there are differences. It also provides some new instructions to align with some of the features that are introduced in the A64 instruction set.

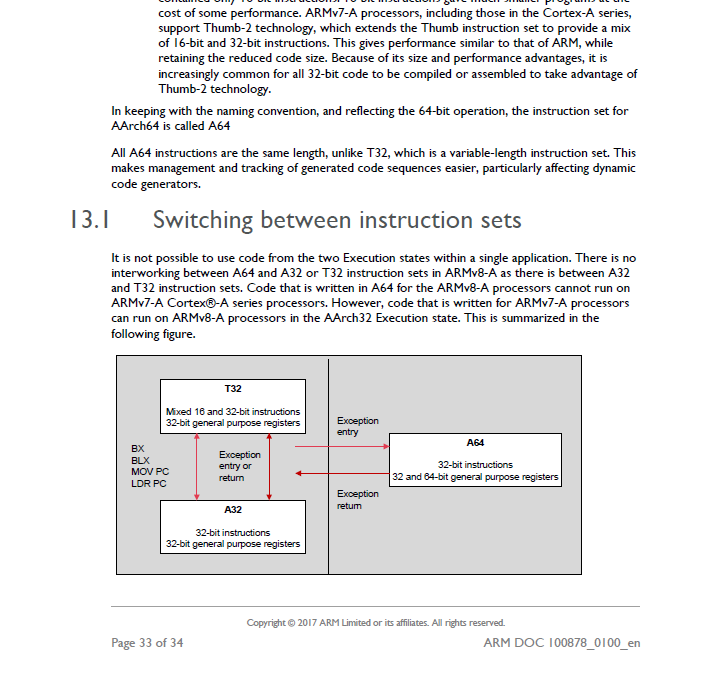
**T32**: The Thumb instruction set was first included in the ARM7TDMI processor and originally contained only 16-bit instructions. 16-bit instructions gave much smaller programs at the cost of some performance. ARMv7-A processors, including those in the Cortex-A series, support Thumb-2 technology, which extends the Thumb instruction set to provide a mix of 16-bit and 32-bit instructions. This gives performance similar to that of ARM, while retaining the reduced code size. Because of its size and performance advantages, it is increasingly common for all 32-bit code to be compiled or assembled to take advantage of Thumb-2 technology.

In keeping with the naming convention, and reflecting the 64-bit operation, the instruction set for AArch64 is called A64.

All A64 instructions are the same length, unlike T32, which is a variable-length instruction set. This makes management and tracking of generated code sequences easier, particularly affecting dynamic code generators.

* Switching between instruction sets

Switching between instruction sets It is not possible to use code from the two Execution states within a single application. There is no interworking between A64 and A32 or T32 instruction sets in ARMv8-A as there is between A32 and T32 instruction sets. Code that is written in A64 for the ARMv8-A processors cannot run on ARMv7-A Cortex®-A series processors. However, code that is written for ARMv7-A processors can run on ARMv8-A processors in the AArch32 Execution state. This is summarized in the following figure.



* Addressing

When the processor can store 64-bit values in a single register, it becomes much simpler to access large amounts of memory within a program. A single thread executing on a 32-bit core is limited to accessing 4GB of address space. Large parts of that addressable space are reserved for use by the OS kernel, library code, peripherals, and more. As a result, lack of space means that the program might need to map some data in or out of memory while executing. Having a larger address space, with 64-bit pointers, avoids this problem. It also makes techniques such as memory-mapped files more attractive and convenient to use. The file contents are mapped into the memory map of a thread, even though the physical RAM might not be large enough to contain the whole file.

* **A32 instruction groups**

| **Instruction group** | **Description** |
| --- | --- |
| Branch and control | These instructions do the following:   * Branch to subroutines. * Branch backwards to form loops. * Branch forward in conditional structures. * Make the following instruction conditional without branching. * Change the processor between A32 state and T32 state. |
| Data processing | These instructions operate on the general-purpose registers. They can perform operations such as addition, subtraction, or bitwise logic on the contents of two registers and place the result in a third register. They can also operate on the value in a single register, or on a value in a register and an immediate value supplied within the instruction.  Long multiply instructions give a 64-bit result in two registers. |
| Register load and store | These instructions load or store the value of a single register from or to memory. They can load or store a 32-bit word, a 16-bit halfword, or an 8-bit unsigned byte. Byte and halfword loads can either be sign extended or zero extended to fill the 32-bit register.  A few instructions are also defined that can load or store 64-bit doubleword values into two 32-bit registers. |
| Multiple register load and store | These instructions load or store any subset of the general-purpose registers from or to memory. |
| Status register access | These instructions move the contents of a status register to or from a general-purpose register. |

* A64 Instruction Set

The A64 instruction set is supported by the Armv8-A architecture. Key features of A64 include:

* Clean decode table based on 5-bit register specifiers.
* Instruction semantics broadly similar to [A32](https://developer.arm.com/architectures/instruction-sets/base-isas/a32) and [T32](https://developer.arm.com/architectures/instruction-sets/base-isas/t32).
* 31 general-purpose 64-bit registers accessible at all times.
* No modal banking of general purpose registers for improved performance and energy.
* Program counter and stack pointer are not general purpose registers.
* Dedicated zero register available for most instructions.
* **A64 instruction groups**

| **Instruction group** | **Description** |
| --- | --- |
| Branch and control | These instructions do the following:   * Branch to and return from subroutines. * Branch backwards to form loops. * Branch forward in conditional structures. * Generate and return from exceptions. |
| Data processing | These instructions operate on the general-purpose registers. They can perform operations such as addition, subtraction, or bitwise logic on the contents of two registers and place the result in a third register. They can also operate on the value in a single register, or on a value in a register and an immediate value supplied within the instruction.  The addition and subtraction instructions can optionally left shift the immediate operand, or can sign or zero-extend and shift the final source operand register.  A64 includes signed and unsigned 32-bit and 64-bit multiply and divide instructions. |
| Register load and store | These instructions load or store the value of a single register or pair of registers from or to memory. You can load or store a single 64-bit doubleword, 32-bit word, 16-bit halfword, or 8-bit byte, or a pair of words or doublewords. Byte and halfword loads can either be sign-extended or zero-extended to fill the 32-bit register. You can also load and sign-extend a signed byte, halfword or word into a 64-bit register, or load a pair of signed words into two 64-bit registers. |
| System register access | These instructions move the contents of a system register to or from a general-purpose register. |

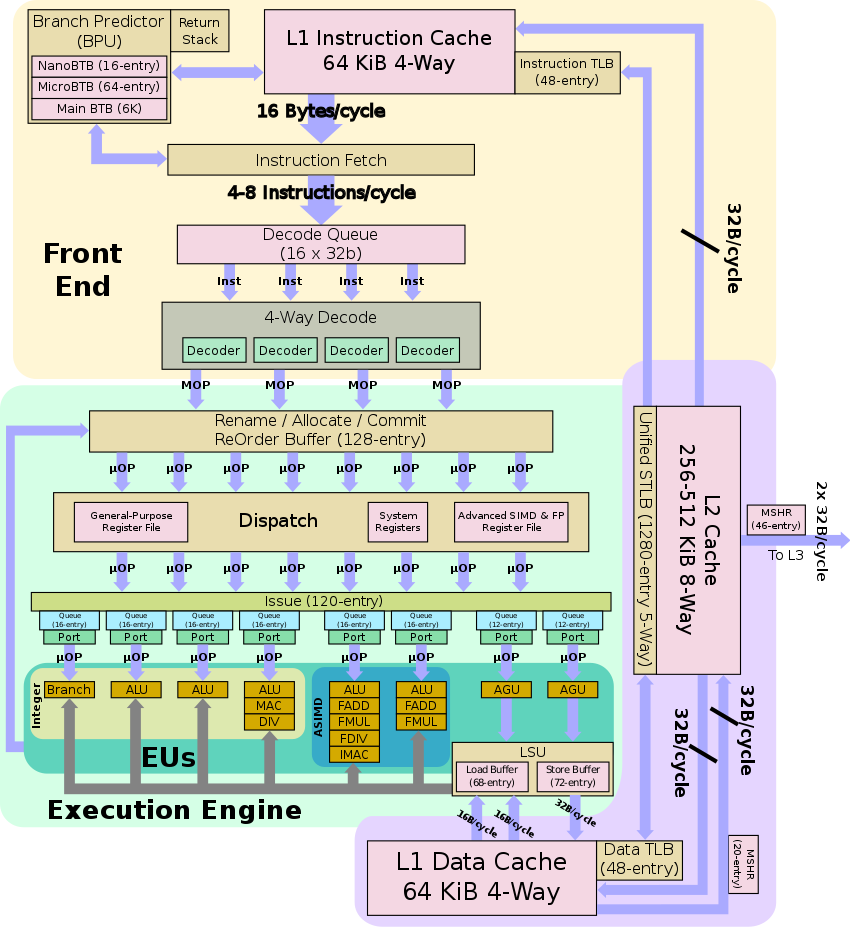
* **Armv8-A Architecture**

The Armv8-A architecture is the latest generation Arm architecture targeted at the Applications ('A') profile.

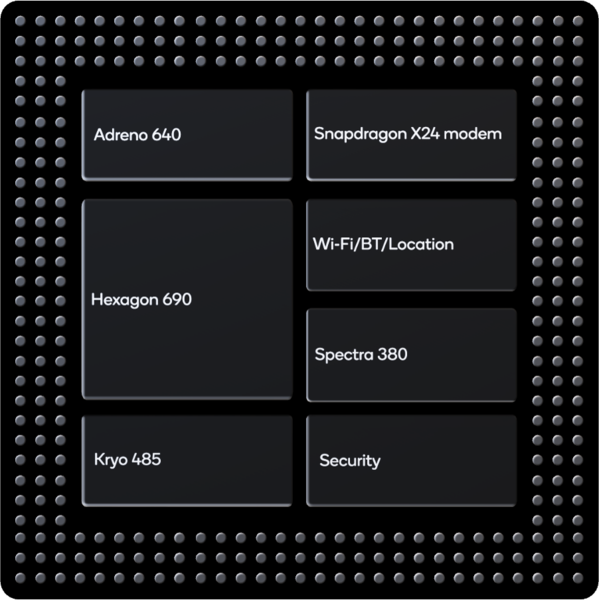
It introduces the ability to use 64-bit and 32-bit Execution states, known as AArch64 and AArch32 respectively. The AArch64 Execution state supports the A64 instruction set, holds addresses in 64-bit registers and allows instructions in the base instruction set to use 64-bit registers for their processing. The AArch32 Execution state is a 32-bit Execution state that preserves backwards compatibility with the Armv7-A architecture and enhances that profile so that it can support some features included in the AArch64 state. It supports the [T32](https://developer.arm.com/architectures/instruction-sets/base-isas/t32) and [A32](https://developer.arm.com/architectures/instruction-sets/base-isas/a32) instruction sets.

Armv8-A is the only profile that supports AArch64 execution, where the relationship between AArch64 and AArch32 is known as *interprocessing*. In addition, the Armv8-A architecture allows different levels of AArch64 and AArch32 support, for example:

* AArch64 only designs.
* AArch64 designs that also support AArch32 operating systems/virtual machines.
* AArch64 support with AArch32 at (unprivileged) application level only.



* **SNAPDRAGON 855 PROCESSOR ARCHITECTURE:**

****

#### Snapdragon 855 supports LP-DDR4X memory type

#### LP-DDR4X

Samsung Semiconductor  proposed an LPDDR4 variant it called LPDDR4X. LPDDR4X is identical to LPDDR4 except additional power is saved by reducing the I/O voltage (Vddq) to 0.6 V from 1.1 V. On 9 January 2017, SK Hynix announced 8 and 6 GiB LPDDR4X packages. JEDEC published the LPDDR4X standard on 8 March 2017 Aside from the lower voltage, additional improvements include a single-channel die option for smaller applications, new MCP, PoP and IoT packages, and additional definition and timing improvements for the highest 4266 Mbit/s speed grade.

* **Memory speed:**2133MHz.

## Functions of the control unit

The Control unit (CU) is digital circuitry contained within the processor that coordinates the sequence of data movements into, out of, and between a processor's many sub-units. The result of these routed data movements through various digital circuits (sub-units) within the processor produces the manipulated data expected by a software instruction (loaded earlier, likely from memory).

It controls (conducts) data flow inside the processor and additionally provides several external control signals to the rest of the computer to further direct data and instructions to/from processor external destinations (i.e. memory).

Examples of devices that require a CU are CPUs and graphics processing units (GPUs). The CU receives external instructions or commands which it converts into a sequence of control signals that the CU applies to the data path to implement a sequence of register-transfer level operations.

More precisely, the Control Unit (CU) is generally a sizable collection of complex digital circuitry interconnecting and directing the many execution units (i.e. ALU, data buffers, registers) contained within a CPU. The CU is normally the first CPU unit to accept from an externally stored computer program a single instruction (based on the CPU's instruction set. The CU then decodes this individual instruction into several sequential steps (fetching addresses/data from registers/memory, managing execution ([i.e. data sent to the ALU or I/O]), and storing the resulting data back into registers/memory) that controls and coordinates the CPU's inner works to properly manipulate the data.

The design of these sequential steps is based on the needs of each instruction and can range in number of steps, the order of execution, and which units are enabled.

Thus by only using a program of set instructions in memory, the CU will configure all the CPU's data flows as needed to manipulate the data correctly between instructions. This results in a computer that could run a complete program and require no human intervention to make hardware changes between instructions (as had to be done when using only punch cards for computations before stored programmed computers with CUs were invented).

These detailed steps from the CU dictate which of the CPU's interconnecting hardware control signals to enable/disable or which CPU units are selected/de-selected and the unit's proper order of execution as required by the instruction's operation to produce the desired manipulated data.

Additionally, the CU's orderly hardware coordination properly sequences these control signals, then configures the many hardware units comprising the CPU, directing how data should also be moved, changed, and stored outside the CPU (i.e. memory) according to the instruction's objective.

Depending on the type of instruction entering the CU, the order and number of sequential steps produced by the CU could vary the selection and configuration of which parts of the CPU's hardware are utilized to achieve the instruction's objective (mainly moving, storing, and modifying data within the CPU).

This one feature, that efficiently uses just software instructions to control/select/configure a computer's CPU hardware (via the CU) and eventually manipulates a program's data, is a significant reason most modern computers are flexible and universal when running various programs. As compared to some 1930s or 1940s computers without a proper CU, they often required rewiring their hardware when changing programs.

This CU instruction decode process is then repeated when the Program Counter is incremented to the next stored program address and the new instruction enters the CU from that address, and so on until the programs end.

Other more advanced forms of Control Units manage the translation of instructions (but not the data containing portion) into several micro-instructions and the CU manages the scheduling of the micro-instructions between the selected execution units to which the data is then channeled and changed according to the execution unit's function (i.e., ALU contains several functions).

On some processors, the Control Unit may be further broken down into additional units, such as an instruction unit or scheduling unit to handle scheduling, or a retirement unit to deal with results coming from the instruction pipeline. Again, the Control Unit orchestrates the main functions of the CPU: carrying out stored instructions in the software program, then directing the flow of data throughout the computer based upon these instructions (roughly likened to how traffic lights will systematically control the flow of cars [containing data] to different locations within the traffic grid (CPU), until it parks at the desired parking spot [memory address/register].

* I/O MECHANISM IN snapdragon 855:

Connectivity:

* X24 LTE modem
  + LTE Category 20
  + Downlink:
    - 2 gbps peak
    - 7x20 MHz carrier aggregation
    - Up to 256-QAM
    - Up to 4x4 MIMO on five carriers
    - Full-Dimension MIMO (FD-MIMO)
    - Maximum 20 spatial streams
  + Uplink:
    - 316 Mbps peak
    - 3x20 MHz carrier aggregation
    - Up to 2x 106Mbps LTE streams
    - Up to 256-QAM
    - Uplink data compression
* LTE FDD, LTE TDD including CBRS support, LAA, LTE Broadcast, WCDMA (DB-DC-HSDPA, DC-HSUPA), TD-SCDMA, CDMA 1x, EV-DO, GSM/EDGE
* Wifi
  + Standards: 802.11ax, 802.11ac Wave 2, 802.11a/b/g, 802.11n
  + Spectral Bands: 2.4 GHz, 5 GHz, 6 GHz
* Bluetooth
  + Bluetooth 5.0
  + 2 Mbps
* **OTHER FUNCTIONALITIES OF DEVICE**:

Features:

1.Support sRGB, Display P3

2.Video Enhancer

3.Reading Mode

4.Night Mode

5.UltraShot

6.Nightscape

7.Studio Lighting

8.Portrait

9.Pro Mode, Panorama, HDR

10.AI Scene Detection