Unit -5

ARCHITECTURES FOR PROGRAMMABLE DSP DEVICES

Architecture of TMS320C5X: Introduction, Bus Structure, Central Arithmetic Logic Unit, Auxiliary Register ALU, Index Register, Block Move Address Register, Parallel Logic Unit, Memory mapped registers, program controller, some flags in the status registers, On- chip memory, On-chip peripherals.

TMS320 Family Overview

The TMS320 family consists of two types of single-chip DSPs:

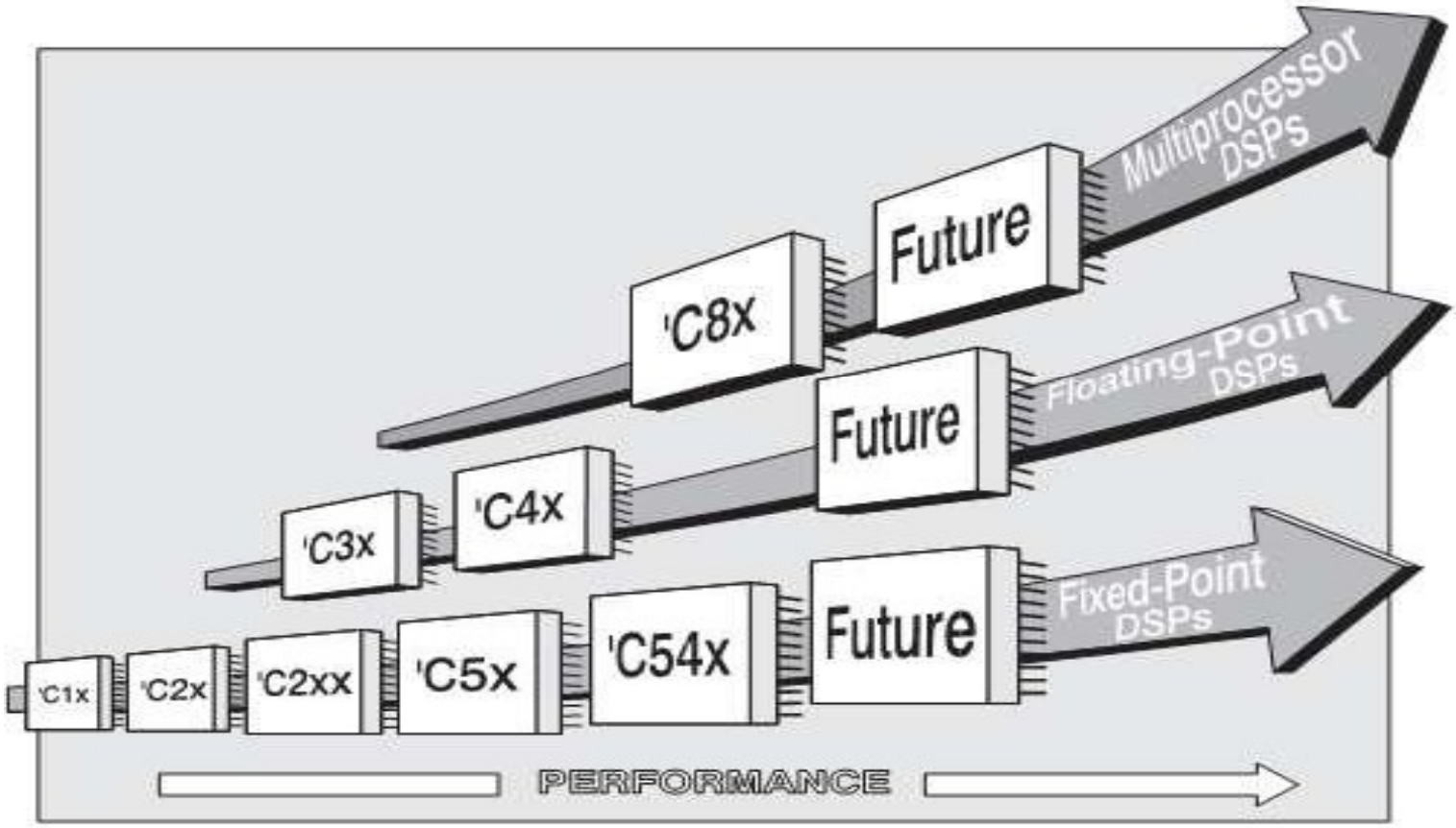
- *16-bit fixed-point* and *32-bit floating-point*. These DSPs possess the **operational flexibility of high-speed controllers** and the **numerical capability of array processors.**

The following characteristics make this family the ideal choice for a wide range of processing applications:

- Very flexible instruction set
- Inherent operational flexibility
- High-speed performance
- Innovative, parallel architectural design
- Cost-effectiveness

Evaluation of the TMS320 family

Figure 1–1. Evolution of the TMS320 Family



TMS320 Overview

- The ’C5x generation consists of the ’C50, ’C51, ’C52, ’C53, ’C53S, ’C56, ’C57, and ’C57S DSPs, which are fabricated by CMOS integrated-circuit technology.
- Their architectural design is based on the ***C25***.
- The ***operational flexibility and speed of the ’C5x*** are the result of combining an advanced ***Harvard architecture*** (which has separate buses for program memory and data memory),
- A CPU with ***application-specific hardware logic, on-chip peripherals, on-chip memory***, and a ***highly specialized instruction set.***
- The ’C5x is designed to execute up to ***50 million instructions per second (MIPS).***

Advantages of TMS320

The ’C5x devices offer these advantages:

- Enhanced TMS320 architectural design for ***increased performance and versatility.***
- Modular architectural design for fast development of ***spin-off devices.***
- Advanced integrated-circuit processing technology for increased performance and ***low power consumption.***
- ***Source code compatibility*** with ’C1x, ’C2x, and ’C2xx DSPs for fast and easy performance upgrades.
- Enhanced ***instruction set for faster algorithms*** and for optimized high-level language operation.
- ***Reduced power consumption*** and increased radiation hardness because of *new static design techniques.*

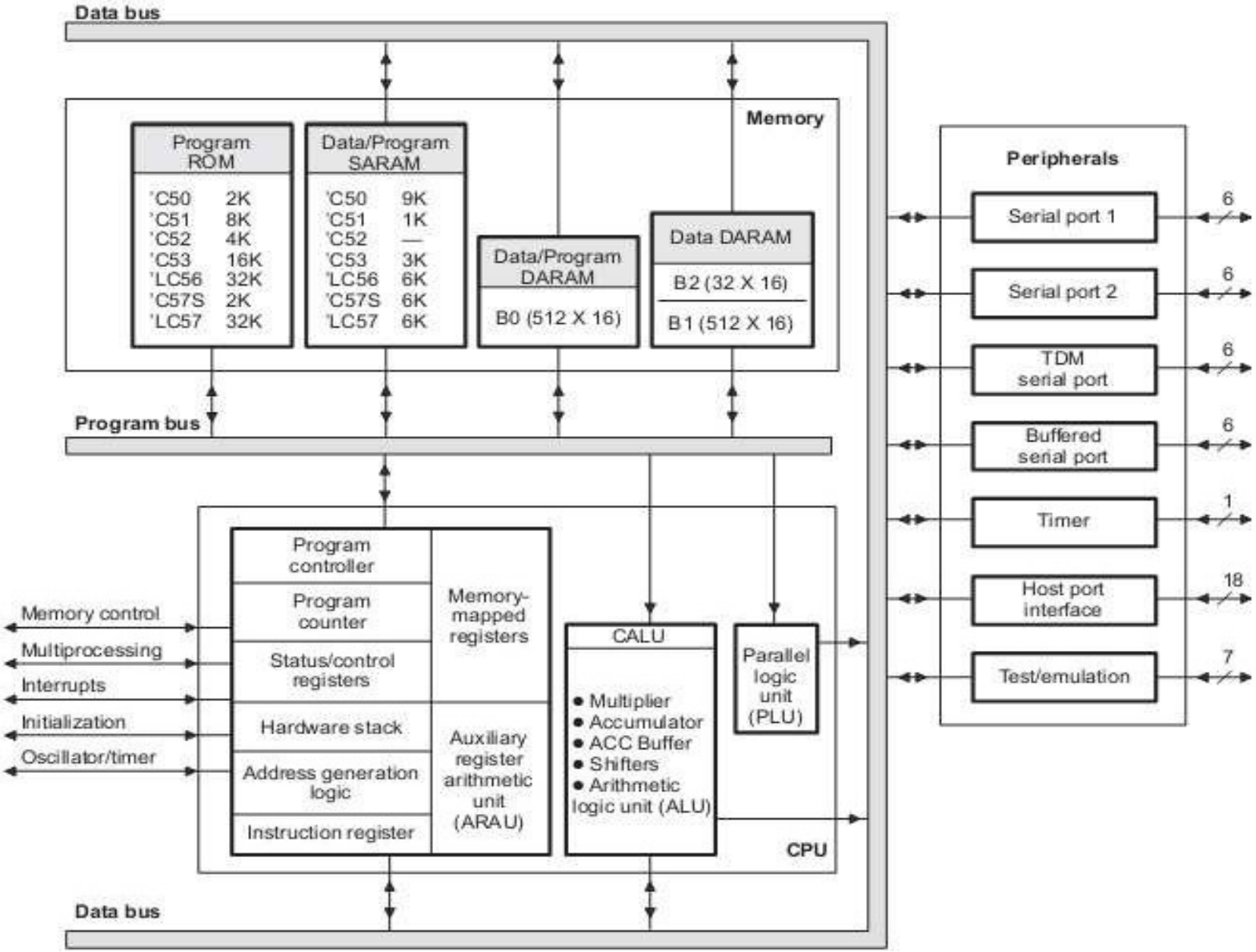
TMS320C5x Key Features

- Compatibility: Source-code compatible with ’C1x, ’C2x, and ’C2xx devices
- Speed: 20-/25-/35-/50-ns single-cycle fixed-point instruction execution time (50/40/28.6/20 MIPS)
- Power
  - 3.3-V and 5-V static CMOS technology with two power-down modes
  - Power consumption control with IDLE1 and IDLE2 instructions for power-down modes

Architectural Overview

- This chapter provides an overview of the architectural structure of the ’C5x,
- which consists of the buses, on-chip memory, central processing unit (CPU), and on-chip peripherals.
- The ’C5x uses an advanced, modified Harvard-type architecture based on the ’C25 architecture and maximizes processing power with separate buses for program memory and data memory.
- The instruction set supports data transfers between the two memory spaces.

Figure 2–1. 'C5x Functional Block Diagram



Bus Structure

- The ’C5x architecture is built around four major buses:
  - Program bus (PB)
  - Program address bus (PAB)
  - Data read bus (DB)
  - Data read address bus (DAB)
    - The ***PAB*** ***provides addresses to program memory space*** for both reads and writes.
    - The ***PB*** also ***carries the instruction code and immediate operands*** from program memory space to the CPU.
    - The ***DB*** ***interconnects various elements of the CPU*** to data memory space.
    - The ***program and data buses*** can work together to transfer data from ***on-chip data memory*** and ***internal or external program memory*** to the ***multiplier for single-cycle multiply/accumulate operations.***

Central Processing Unit (CPU)

- The ’C5x CPU consists of these elements:
  - Central arithmetic logic unit (CALU)
  - Parallel logic unit (PLU)
  - Auxiliary register arithmetic unit (ARAU)
  - Memory-mapped registers
  - Program controller

Central Arithmetic Logic Unit (CALU)

The CPU uses the CALU to perform 2s-complement arithmetic. The CALU consists of these elements:

- 16-bit x 16 -bit multiplier
- 32-bit arithmetic logic unit (ALU)
- 32-bit accumulator (ACC)
- 32-bit accumulator buffer (ACCB)
- Additional shifters at the outputs of both the accumulator and the product register (PREG)

Parallel Logic Unit (PLU)

- The CPU includes an independent PLU, which operates separately from, but in parallel with, the ALU.
- The PLU performs Boolean operations or the bit manipulations required of high-speed controllers.
- The PLU can *set, clear, test, or toggle* bits in a status register, control register, or any data memory location.
- The PLU provides a *direct logic operation* path to data memory values without affecting the contents of the ACC or PREG.

Auxiliary Register Arithmetic Unit (ARAU)

- The CPU includes an ***unsigned 16-bit arithmetic logic unit*** that calculates ***indirect addresses by using inputs from the auxiliary registers (ARs), index register (INDX), and auxiliary register compare register (ARCR).***
- The ***ARAU can auto index the current AR*** while the data memory location is being addressed and can index either by +/- 1 or by the contents of the INDX.
- As a result, ***accessing data does not require the CALU for address manipulation; therefore, the CALU is free for other operations in parallel***

Memory-Mapped Registers

- The memory-mapped registers are used for ***indirect data address pointers, temporary storage, CPU status and control, or integer arithmetic processing through the ARAU.***

- Since the memory-mapped registers are a component of the data memory space, they can be written to and read from in the same way as any other data memory location.

Program Controller

- ***The program controller consists of these elements:***
  - ❑ Program counter
  - ❑ Status and control registers
  - ❑ Hardware stack
  - ❑ Address generation logic
  - ❑ Instruction register

❖The program controller contains ***logic circuitry that decodes the operational instructions, manages the CPU pipeline, stores the status of CPU operations, and decodes the conditional operations.***

On-Chip Memory

- The ’C5x architecture contains a considerable amount of on-chip memory to aid in system performance and integration:
  - ❑ ***Program read-only memory (ROM)***
  - ❑ ***Data/program dual-access RAM (DARAM)***
  - ❑ ***Data/program single-access RAM (SARAM)***

➢ The ’C5x has a total address range of ***224K words X 16 bits.*** The memory space is divided into four individually selectable memory segments:

Memory space is divided into four individually selectable memory segments:

- ✓ **64K-**word program memory space,
- ✓ **64K**-word local data memory space,
- ✓ **64K-**word input/ output ports,
- ✓ 32K-word global data memory space.

1. **Program ROM :** This memory is used for ***booting program code*** from **slower external ROM or EPROM** to **fast on-chip or external RAM.**
2. **Data/Program Dual-Access RAM :** All ’C5x DSPs carry a 1056-word X 16-bit on-chip dual-access RAM (DARAM).

The DARAM is divided into three individually selectable memory blocks:

- 512-word data or program DARAM block B0,
- 512-word data DARAM block B1,
- 32-word data DARAM block B2.

The DARAM is ***primarily intended to store data values*** but, when needed, can be ***used to store programs as well.***

DARAM improves the ***operational speed*** of the ’C5x CPU as The CPU operates with a ***4-deep pipeline.***

3. Data/Program Single-Access RAM :

- All ’C5x DSPs except the ’C52 carry a 16-bit on-chip single-access RAM (SARAM) of various sizes
- Code can be ***booted from an off-chip ROM*** and ***then executed at full speed, once it is loaded into the on-chip SARAM.***

The SARAM can be configured by software in one of three ways:

- All SARAM configured as ***data memory***
- All SARAM configured as ***program memory***
- SARAM configured as both ***data memory and program memory***

- **On-Chip Memory Protection :**
  - The ’C5x DSPs have a ***maskable option*** that protects the contents of on-chip memories. When the related bit is set, no externally originating instruction can access the on-chip memory spaces.
- **On-Chip Peripherals :** All ’C5x DSPs have the same CPU structure; however, they have different on-chip peripherals connected to their CPUs. The ’C5x DSP on-chip peripherals available are:
  - **Clock generator**
  - **Hardware timer**
  - **Software-programmable wait-state generators**
  - **Parallel I/O ports**
  - **Host port interface (HPI)**
  - **Serial port**
  - **Buffered serial port (BSP)**
  - **Time-division multiplexed (TDM) serial port**
  - **User-maskable interrupts**

Peripherals

1. **Serial Port :** Three different kinds of serial ports are available:
   - a general-purpose serial port,
   - a time-division multiplexed (TDM) serial port,
   - a buffered serial port (BSP).
     - Each ’C5x contains at ***least one general-purpose, high-speed synchronous, full-duplexed serial port interface that provides direct communication with serial devices such as codecs, serial analog-to-digital (A/D) converters, and other serial systems.***
     - The serial port is **capable of operating** at up to **one-fourth the machine cycle rate** (CLKOUT1).
     - The ***serial port transmitter and receiver are double-buffered*** and ***individually controlled by maskable external interrupt signals.*** Data is framed either as **bytes** or as **words.**

2. Buffered Serial Port (BSP):

- The BSP available on the ’C56 and ’C57 devices is a **full-duplexed, double-buffered serial port** and **an auto buffering unit (ABU).**
- The ABU supports ***high-speed data transfer*** and ***reduces interrupt latencies.***

3. TDM Serial Port:

- The TDM serial port available on the ’C50, ’C51, and ’C53 devices is a ***full-duplexed serial port***
- that can be configured by ***software*** either for ***synchronous operations*** or for ***time-division multiplexed*** operations.
- The TDM serial port is commonly used in ***multiprocessor applications.***

4. User-Maskable Interrupts:

- **Four *external interrupt lines (INT1 –INT4)***
- **Five *internal interrupts***,
- A ***timer interrupt*** and
- **Four *serial port interrupts***, are user maskable.

5. Test/Emulation:

- On the ’C50, ’LC50, ’C51, ’LC51, ’C53, ’LC53, ’C57S and ’LC57S, ***an IEEE standard 1149.1*** (JTAG) interface with ***boundary scan capability*** is ***used for emulation and test***

6. Clock Generator: The clock generator consists of an ***internal oscillator and a phase-locked loop (PLL)*** circuit. The clock generator can be ***driven internally by a crystal resonator circuit*** or ***driven externally by a clock source***

7. Hardware Timer: A 16-bit hardware timer with a 4-bit pre-scaler is available. The timer can be ***stopped, restarted, reset, or disabled*** by **specific status bits.**

8. Software-Programmable Wait-State Generators: Software-programmable wait-state logic is incorporated in 'C5x DSPs allowing wait-state generation without any external hardware for interfacing with slower off-chip memory and I/O devices.

9. Parallel I/O Ports: A total of ***64K I/O ports are available, sixteen*** of these ports are memory-mapped in data memory space. Each of the I/O ports can be addressed by the ***IN or the OUT instruction.***

10. Host Port Interface (HPI): The HPI available on the ***'C57S and 'LC57 is an 8-bit parallel I/O port*** that pro-vides an interface to a ***host processor.***

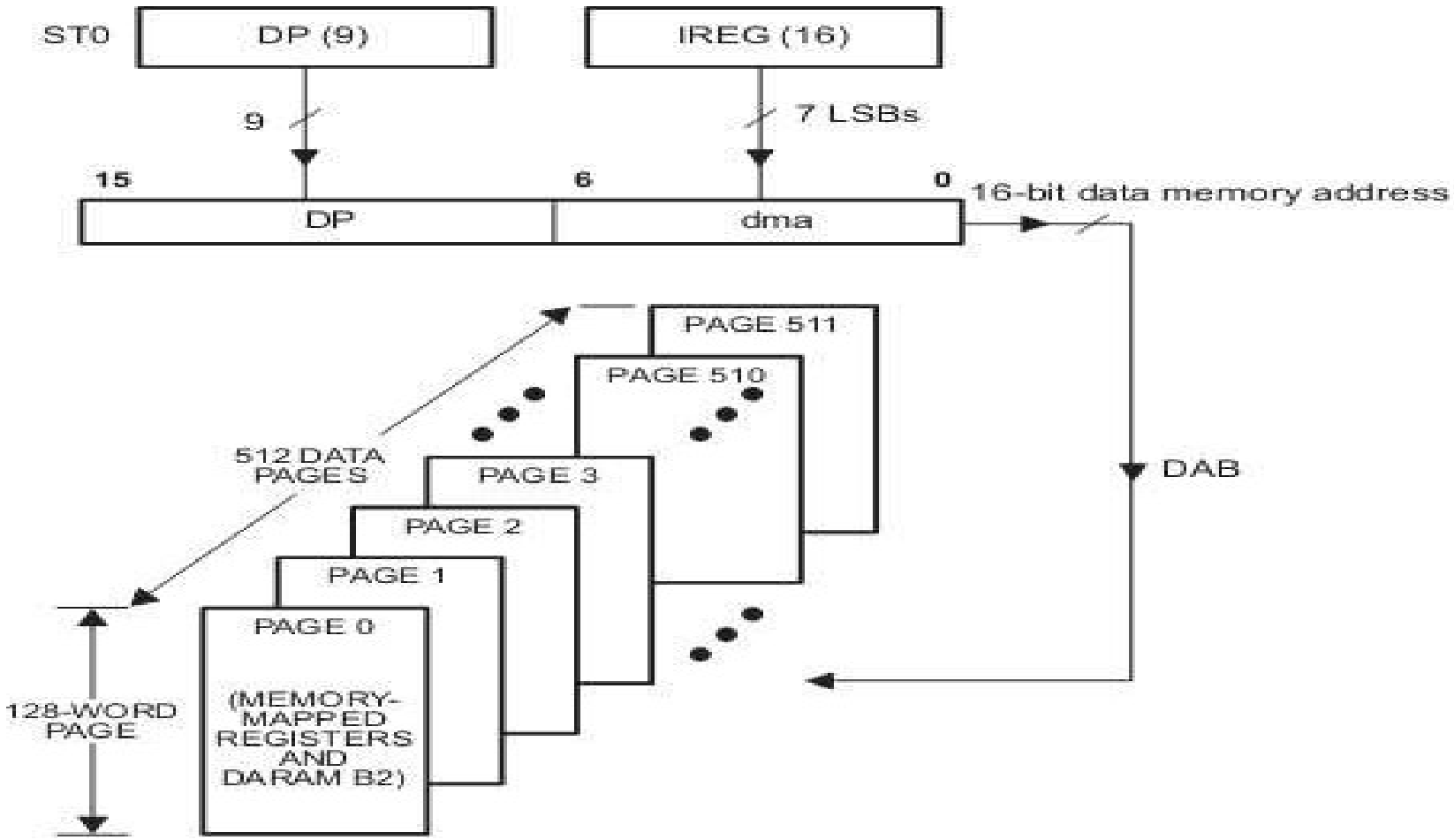
Addressing Modes

- Direct addressing
- Indirect addressing
- Immediate addressing
- Dedicated-register addressing
- Memory-mapped register addressing
- Circular addressing

1.Direct Addressing

- In the direct memory addressing mode, the instruction contains the ***lower 7 bits of the data memory address (dma).***
- The 7-bit dma is concatenated with ***the 9 bits of the data memory page pointer (DP) in status register 0*** to form the full 16-bit data memory address.
- This ***16-bit data memory address*** is placed on an internal direct data memory address bus ***(DAB).***
- The DP points to one of 512 possible data memory pages and the 7-bit address in the instruction points to one of 128 words within that data memory page.
- You can load the DP bits by using the LDP or the LST #0 instruction.

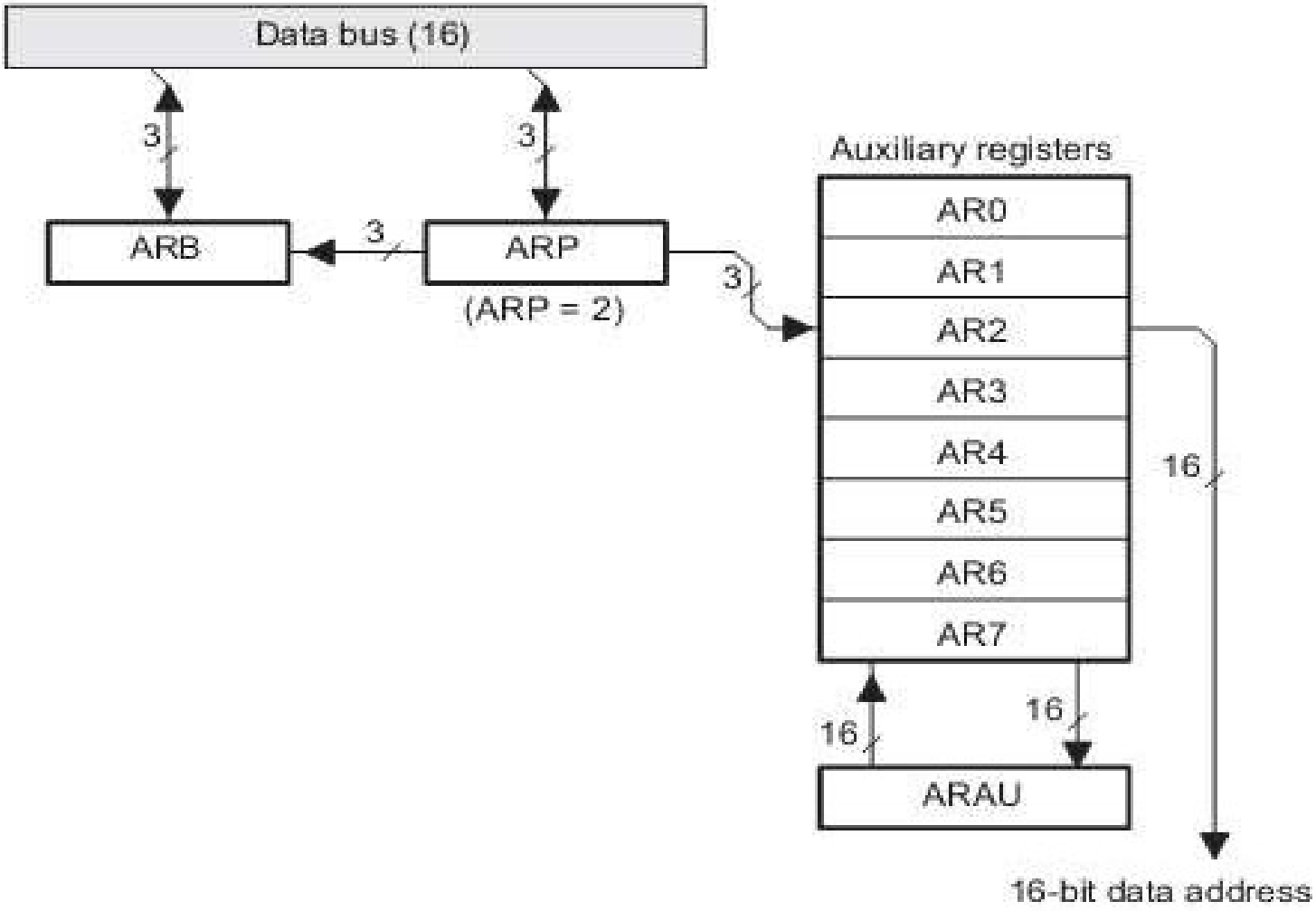
Direct Addressing…



2.Indirect Addressing

- ***Eight 16-bit auxiliary registers (AR0–AR7)*** provide flexible and powerful indirect addressing.
- In indirect addressing, any location in the ***64K-word data memory*** space can be accessed using a ***16-bit address*** contained in an AR.
- Figure shows the hardware for indirect addressing.

Indirect Addressing….



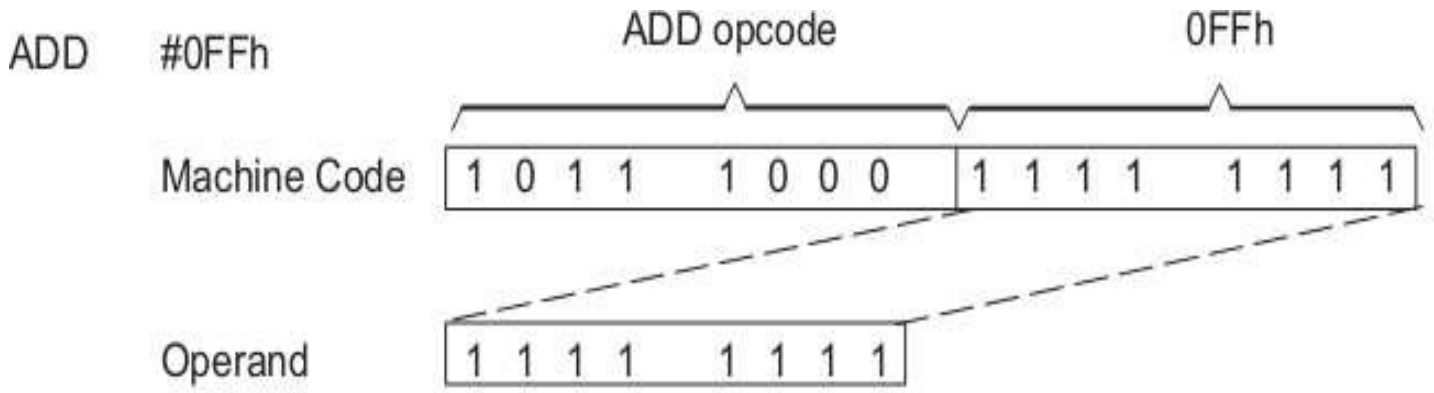
- To select a specific AR, load the auxiliary register pointer (ARP) with ***a value from 0 through 7,*** designating AR0 through AR7, respectively.
- The register pointed to by the ARP is referred to as the current auxiliary register (current AR).
- You can load the address into the AR using the LAR instruction

3.Immediate Addressing

- In immediate addressing, the instruction word(s) contains the value of the ***immediate operand.*** The ’C5x has both 1-word (8-bit, 9-bit, and 13-bit constant) short immediate instructions
- 2-word (16-bit constant) long immediate instructions.
- This mode is indicated by symbol *# .for e.g*
- *ADD # 56h : adds 56h to ACC.*
- *ADD # 4567h : adds 4567h to ACC*

3.1.Short Immediate addressing

- In short immediate instructions, the operand is contained within the instruction machine code. Figure shows an example of the short immediate mode.
- Note that in this example, the lower 8 bits are the operand and will be added to the ACC by the CALU.



3.2.Long Immediate Addressing

- In long immediate instructions, the operand is contained in ***the second word of a two-word instruction.*** There are two long immediate addressing modes:

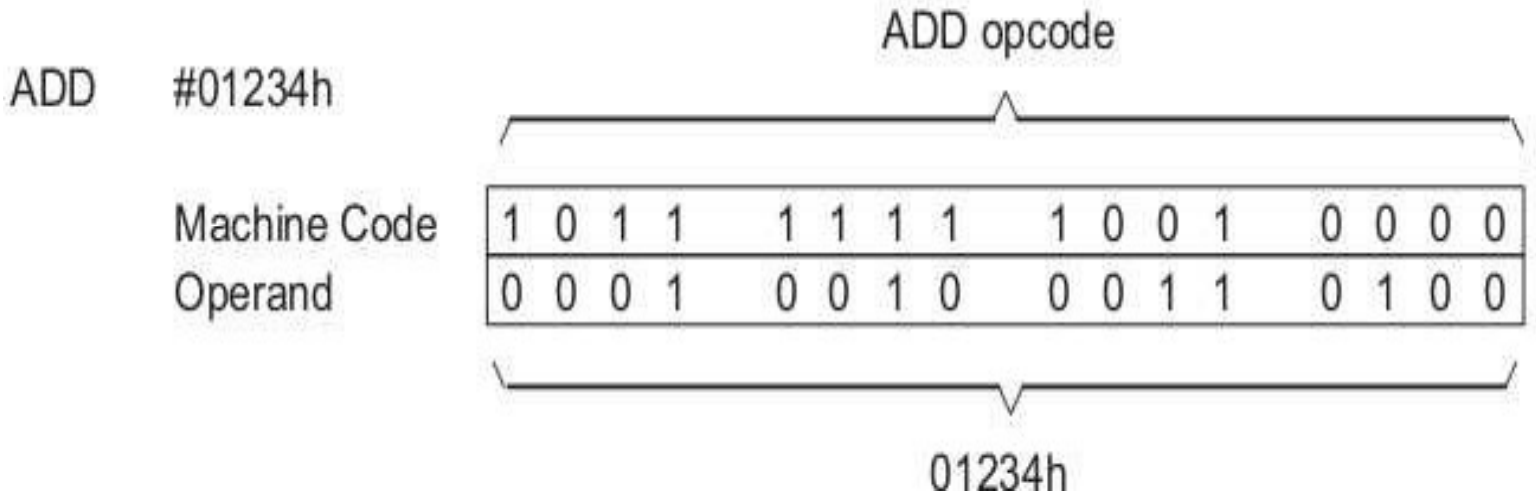
➢ One-operand instructions

➢ Two-operand instructions

Long Immediate Addressing with Single/No Data memory Access :

Figure shows an example of long immediate addressing with no data memory access. In Figure the second word of the 2-word instruction is added to the ACC by the CALU**.**

Long Immediate Addressing Mode — No Data Memory Access



4.Dedicated-Register Addressing

- The dedicated-registered addressing mode operates like the long immediate addressing mode,
- Where address comes from one of two special-purpose memory-mapped registers in the CPU:

➢ The block move address register (BMAR)

➢ The dynamic bit manipulation register (DBMR).

The advantage of this addressing mode is that the address of the block of memory to be acted upon can be changed during execution of the program.

- The syntax for dedicated-register addressing can be stated in one of two ways:

❑ Specify BMAR by its predefined symbol:

```
ELDD BMAR,DAT100;DP = 0. BMAR contains the value 200h.
```

The content of data memory location 200h is copied to data memory location 100 on the current data page.

❑ Exclude the immediate value from a parallel logic unit (PLU) instruction:

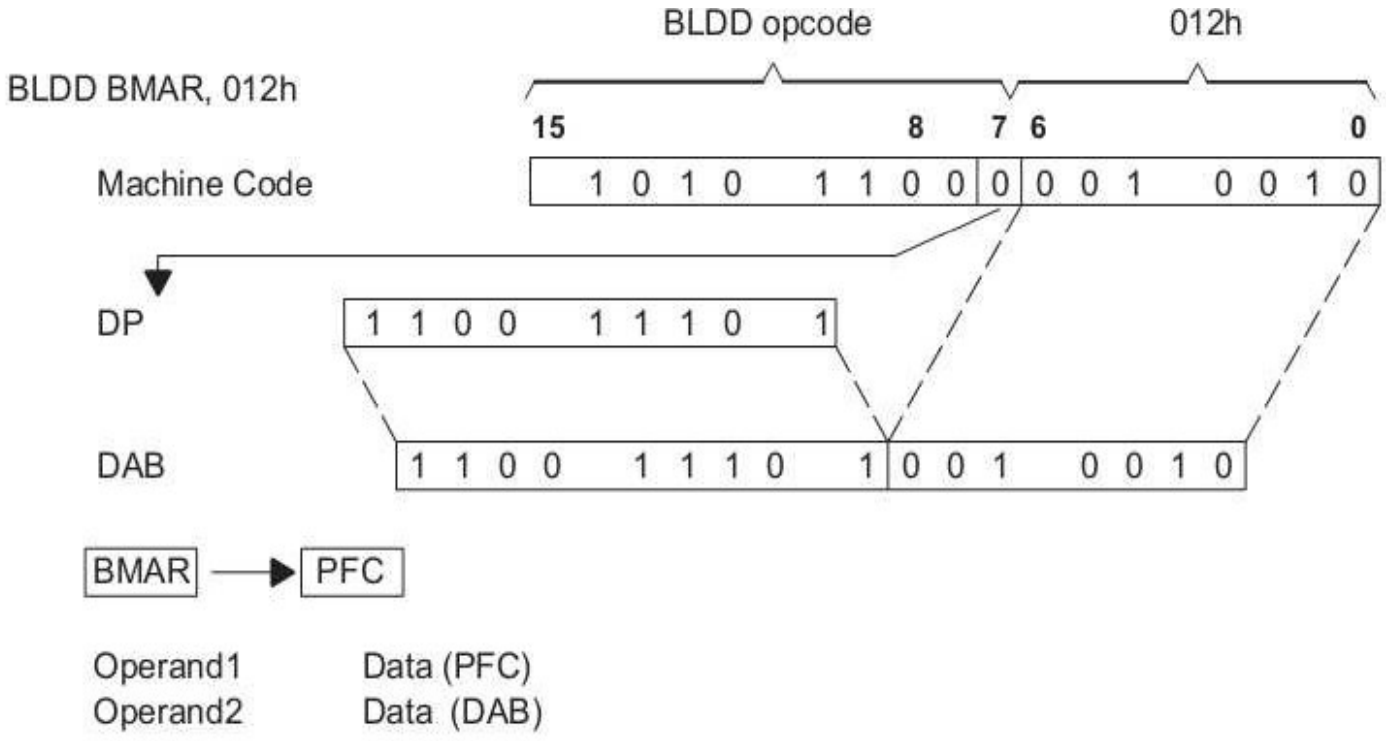
```
OPL DAT10;DP = 6. DBMR contains the value FFF0h; Address 030Ah contains the value 01h
```

The content of data memory location 030Ah is ORed with the content of the DBMR. The resulting value FFF1h is stored back in memory location
030Ah.

4.1.Using the Contents of the BMAR

- The BLDD, BLDP, and BLPD instructions use the BMAR to point at the source or destination space of a block move.
- The MADD and MADS instructions also use the BMAR to address an operand in program memory for a multiply-accumulate operation.

Figure shows how the BMAR is used in the dedicated-register addressing mode. **Bits 15 through 8** of the machine code contain the opcode. **Bit 7, with a value of 0,** defines the addressing mode as direct, and **bits 6 through 0** contain the dma.

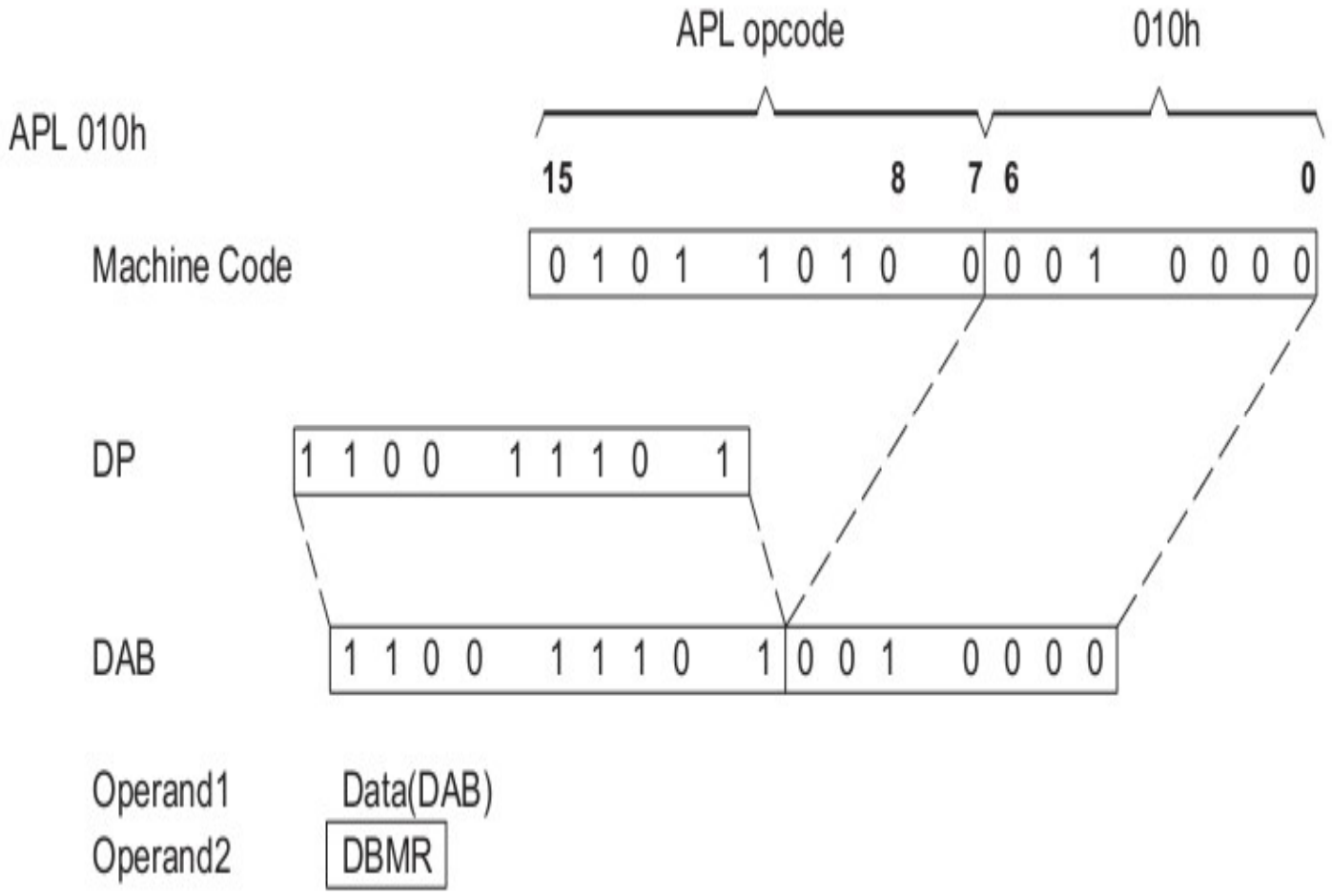


Note: DAB is the 16-bit internal data memory address bus.

4.2.Using the Contents of the DBMR

- The APL, CPL, OPL, and XPL instructions use the PLU and the contents of the DBMR when an immediate value is not specified as one of the operands.
- Figure illustrates how the DBMR is used as an AND mask in the APL instruction.
- **Bits 15 through 8** of the machine code contain the opcode.
- **Bit 7,with a value of 0,** defines the addressing mode as direct, and **bits 6 through 0** contain the dma.

Figure. Dedicated-Register Addressing Using the DBMR



Note: DAB is the 16-bit internal data memory address bus.

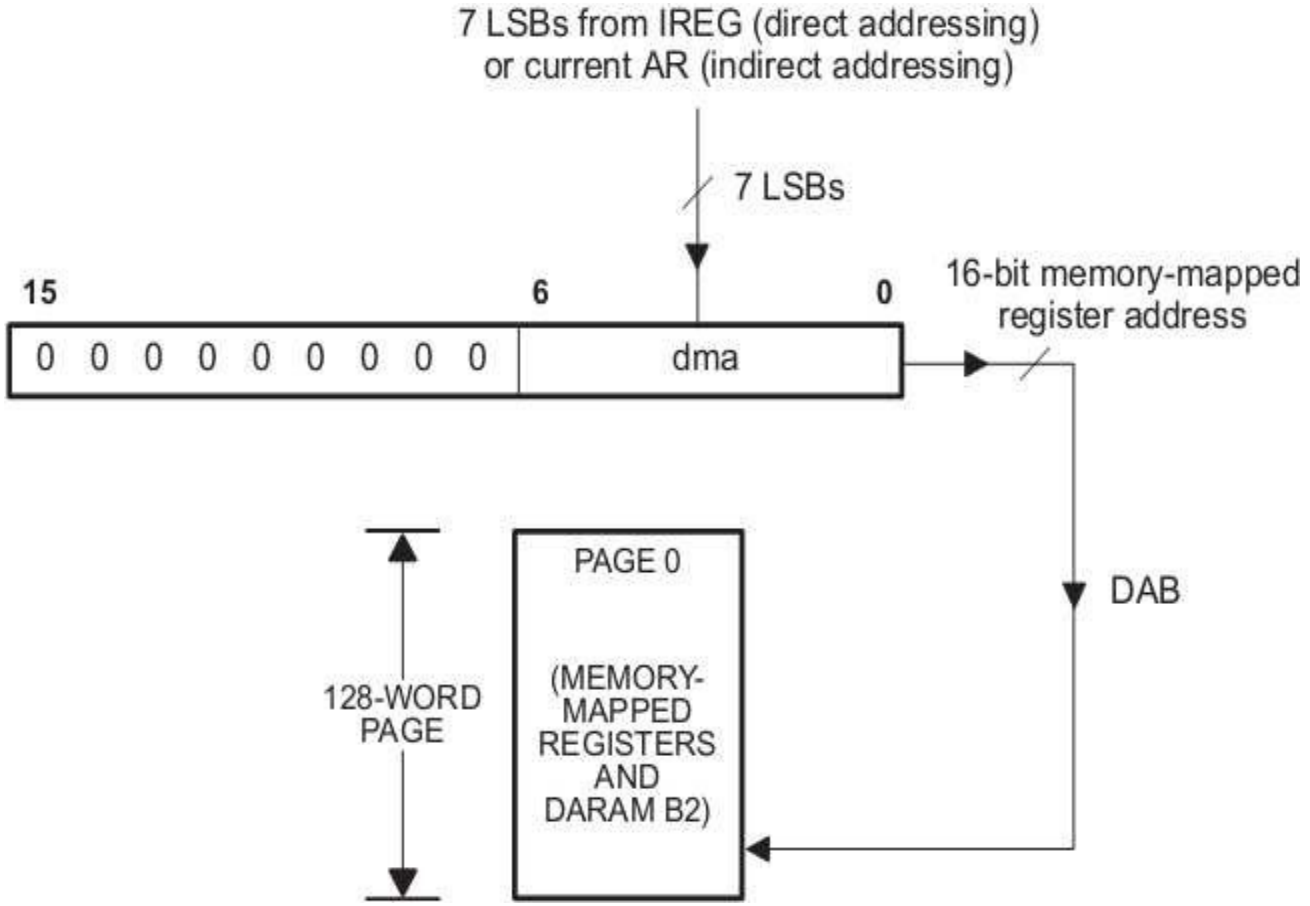
5.Memory-Mapped Register Addressing

- With memory-mapped register addressing, ***you can modify the memory-mapped registers without affecting the current data page pointer value.***
- In addition, ***you can modify any scratch pad RAM (DARAM B2) location or data page 0.***
- The memory-mapped register addressing mode operates like the ***direct addressing mode,*** except that the 9 MSBs of the address are forced to 0 instead of being loaded with the contents of the DP.
- This allows you to ***address the memory-mapped registers of data page 0 directly without the overhead of changing the DP or auxiliary register.***

- The following instructions operate in the memory-mapped register addressing mode. Using these instructions does not affect the contents of the DP:
  - ❑ LAMM — Load accumulator with memory-mapped register
  - ❑ LMMR — Load memory-mapped register
  - ❑ SAMM — Store accumulator in memory-mapped register
  - ❑ SMMR — Store memory-mapped register

  Figure illustrates how this is done by forcing the 9 MSBs of the data memory address to 0, regardless of the current value of the DP when direct addressing is used or of the current AR value when indirect addressing is used.

Fig: Memory-Mapped Register Addressing



6.Circular Addressing

- Many algorithms such as ***convolution, correlation, and finite impulse response (FIR) filters*** can use circular buffers in memory to implement a sliding window, which contains the most recent data to be processed.
- The ’C5x supports two concurrent circular buffers operating via the ARs.
- The following five memory-mapped registers control the circular buffer operation:
  - **CBSR1 — Circular buffer 1 start register**
  - **CBSR2 — Circular buffer 2 start register**
  - **CBER1 — Circular buffer 1 end register**
  - **CBER2 — Circular buffer 2 end register**
  - **CBCR — Circular buffer control register**

1. To define circular buffers, you first load the start and end addresses into the corresponding buffer registers;
2. Load a value between the start and end
3. Registers for the circular buffer into an AR.
4. Load the proper AR value, and set the corresponding circular buffer enable bit in the CBCR.