Computer Architecture Lab Session 11

# Basic Computer

# Bardia Ardakanian 9831072

# Ali Asad 9831004

# توضیح کلی

در این گزارش می‌خواهیم کد کامپیوتر پایه داده شده را تحلیل کنیم.

هر BC از یک مسیر داده و واحد تصمیم گیری تشکیل می‌شود.

مسیر داده یا Datapath در واقع مسیری است که داده های موجود درثبات ها با آن تبادل اطلاعات می‌کنند.

و واحد تصمیم‌گیری یا Control Unit نیز به عنوان کنترل کننده تمامی ثبات ها به کار برده می‌شود.

به این معنی که هر ثبات که شامل پایه های مختلف از قبیل Load,Clear,Inc,Dec,… هر کدام چه زمانی با توجه به الگوریتم ون نیومن صدا زده شود و چه زمانی داده های ثبات ها بر روی گذرگاه ریخته شود و چه زمانی داده ها از گذرگاه به ثبات ها وارد شود.

از دیدگاهی دیگر میتوان کامپیوتر پایه را متشکل از یک حافظه اصلی و یک پردازنده دید که دستورالعمل ها از حافظه اصلی به پردازنده میرود و پردازنده بعد از اجرا آن طی الگوریتم ون نیومن، درصورت نیاز، نتیجه را در حافظه اصلی می‌ریزد.

# ورودی و خروجی

این ماژول ورودی‌هایی دارد که به توضیح هر یک می‌پردازیم

entity microprocessor is  
 port  
 (  
 reset :in std\_logic; -- reset  
 clk :in std\_logic; -- clock  
 data :inout std\_logic\_vector(15 downto 0); -- data lines  
 address :out std\_logic\_vector(11 downto 0); -- address lines  
 memr :out std\_logic; -- memory read  
 memw :out std\_logic; -- memory write  
 inport :in std\_logic\_vector(7 downto 0); -- input port  
 outport :out std\_logic\_vector(7 downto 0); -- output port  
 intr\_in :in std\_logic; -- interrupt for input  
 intr\_out :in std\_logic -- interrupt for output  
 );  
end microprocessor;

## حافظه اصلی

با توجه به ورودی‌ها متوجه می‌شویم هر ادرس از 12 بیت تشکیل شده است. این بدان معناست که حافظه اصلی ما از 4096 کلمه (word) تشکیل شده است.

همچنین داده (Data) ورودی ما 16 بیتی است، این بدان معناست که هر کلمه در حافظه اصلی از 16 بیت تشکیل شده.

در نتیجه حافظه اصلی ما، یک حافظه 4 کیلوبایتی با کلمه (word) های 16 است.

## read و write حافظه اصلی

برای آنکه از حافظه اصلی داده برداریم یا به آن داده بریزیم باید پایه‌های read و write آن را مشخص کنیم تا فقط در زمانی که باید و پایه‌ها فعال هستند، این خواندن و نوشتن روی حافظه اصلی اتفاق بیافتد.

## ادرس حافظه

همانطور که گفته شد حافظه اصلی ما 4 کیلوبایتی است و برای مشخص کردن هر کلمه در آن، به 12 بیت احتیاج داریم.

## پورت ورودی و خروجی

پورت ورودی و خروجی، داده را از خارج ریزپردازنده گرفته و یا به خارج ریزپردازنده ارسال می‌کنند.

## وقفه

دو پورت وقفه ورودی و خروجی داریم که کاربردشان گرفتن وقفه و دادن interrupt acknowledge است.

## پورت reset و clk

پایه ریست و کلاک پردازنده.

# ثبات های عام منظوره و خاص منظوره

بیان شد که هر پردازنده تعدادی ثبات دارند.

این ثبات ها به دو دسته ثبات های عام منظوره و ثبات های خاص منظوره تقسیم می‌شوند.

ثبات های عام منظوره همانطور که از نامشان معلوم است، بر خلاف ثبات های خاص منظوره که برای هدف و منظور خاصی طراحی شده اند، می‌توانند توسط اجزای مختلف به کار گرفته شود.

# عملکرد ثبات های پردازنده

حال به بیان ثبات های مختلف میپردازیم:

PC: Program Counter

نگه داشتن آدرس دستورالعمل بعدی که باید اجرا بشود.

AR: Address Register

رهگیری محلی از مموری که در حال اجرا است. به بیان دیگر در خود آدرس ذخیره میکند.

AC: Accumulator

ثبات عام منظور ای به نام انباشتگر است.

TR: Temp Register

ثباتی برای بیان آدرس ابتدا پشته در حافظه اصلی

inpr: Input Register

ثباتی برای نگه داری ورودی( برای دستورات IO استفاده میشود)

outr: Output Register

ثباتی برای نگه داری خروجی( برای دستورات IO استفاده میشود)

DR: Data Register

ثباتی برای ذخیره عملوند های دستورالعمل

در این ریزپردازنده ثبات‌های فوق وجود دارند. اندازه هر ثبات بسته به کاربرد آن متفاوت است.

به عنوان مثال:

ثبات‌های PC و AR چون آدرس حافظه در خود نگه می‌دارند باید 12 بیتی باشند.

ثبات‌های DR, TR, AC چون محتویات یک کلمه (word) حافظه را در خود نگه می‌دارند 16 بیتی هستند.

ثبات IR چون دستورالعمل در خود نگه میدارد ضریبی از کلمه (word) است. این ضریب بسته به ساختار دستورالعمل متفاوت است. در این ریزپردازنده اندازه ثبات IR 16 بیتی است.

ثبات‌های itpr, outr چون محتویات پورت ورودی و خروجی را در خود نگه می‌دارندر 8 بیتی هستند.

|  |  |
| --- | --- |
| اندازه ثبات | نام ثبات |
| 16 | DR |
| 12 | AR |
| 16 | AC |
| 12 | IR |
| 12 | PC |
| 16 | TR |
| 8 | INPR |
| 8 | OUTR |

# سیگنال‌های داخلی

هر ریزپردازنده سیگنال‌های داخلی دارد که در انجام دستورالعمل‌های مختلف از انها استفاده می‌کند.

لیست این سیگنال‌ها در جدول زیر آمده:

|  |  |  |
| --- | --- | --- |
| کربرد | اندازه سیگنال | نام |
| instruction decoder output | 8 | d |
| sequence counter output | 4 | sc |
| timing signals | 16 | t |
| I bit | 1 | i |
| extended accumulator | 1 | e |
| start-stop flip-flop | 1 | s |
| enable signal for instruction decoder | 1 | en\_id |
| clear signal for sequence counter | 1 | clr\_sc |
| input flag | 1 | fgi |
| output flag | 1 | fgo |
| interrupt enable flip-flop | 1 | ien |
| interrupt flip-flop | 1 | r |

بعضی از این سیگنال‌ها پایه‌های ثبات‌ها و یا پایه enable دیکودر هستند. بعضی دیگر هم flag هستند. Flag‌ها را با رنگ ابی و پایه ثبات‌ها و دیکودر با رنگ نارنجی مشخص شده‌اند.

و همینطور سیگنال d، دیکود شده opcode دستورالعمل ورودی است.

# قالب دستورالعمل

تبادل دستورالعمل ها در قالب هایی از پیش تعیین شده بیان می‌شود تا پردازنده بفهمد که رشته بیتی که از حافظه اصلی دریافت کرده است را چگونه تفسیر کند.

این قالب دستورالمعل با توجه به ISA یا به طور واضح تر،‌ عملیات هایی که میخواهیم انجام بدهیم مشخص می‌شود.

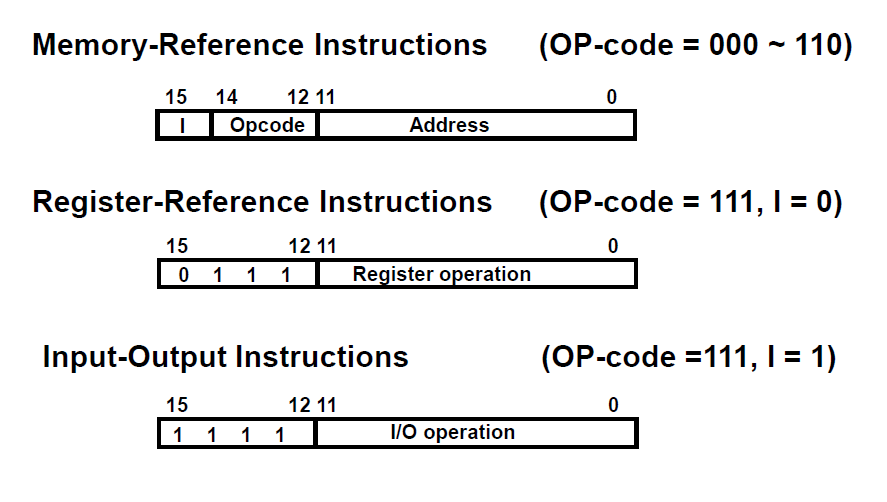
اما به طور کلی در دستورالعمل ها چند بیت برای OPC که مشخص کننده عملیات موردنظر است، درنظر گرفته می‌شود.

و نیز شامل رشته بیتی برای مشخص کردن آدرس مموری و همچنین شامل چند بیت برای تشخیص

ثبات عام منظوره است.

همانطور که گفته شد، موارد گفته شده کلی بوده و با توجه به عملیات های موردانتظار در CPU مشخص می‌شود.

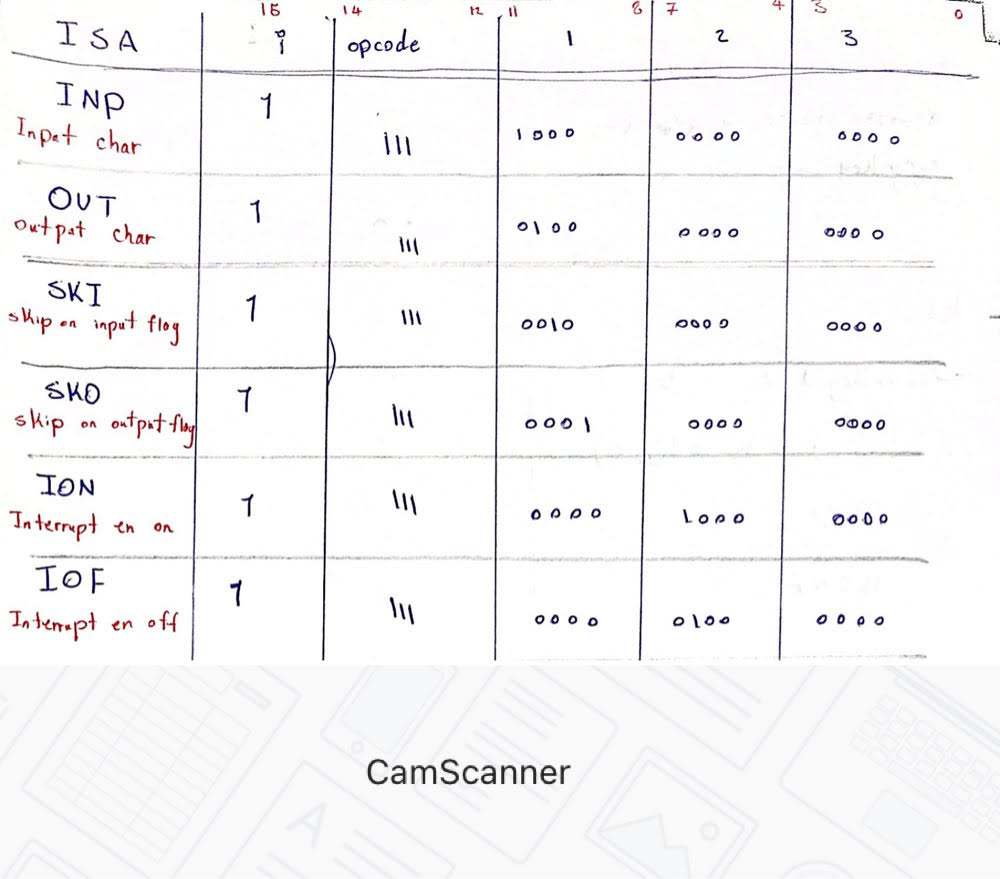
در این ریزپردازنده قالب دستورعمل به شرح زیر است:



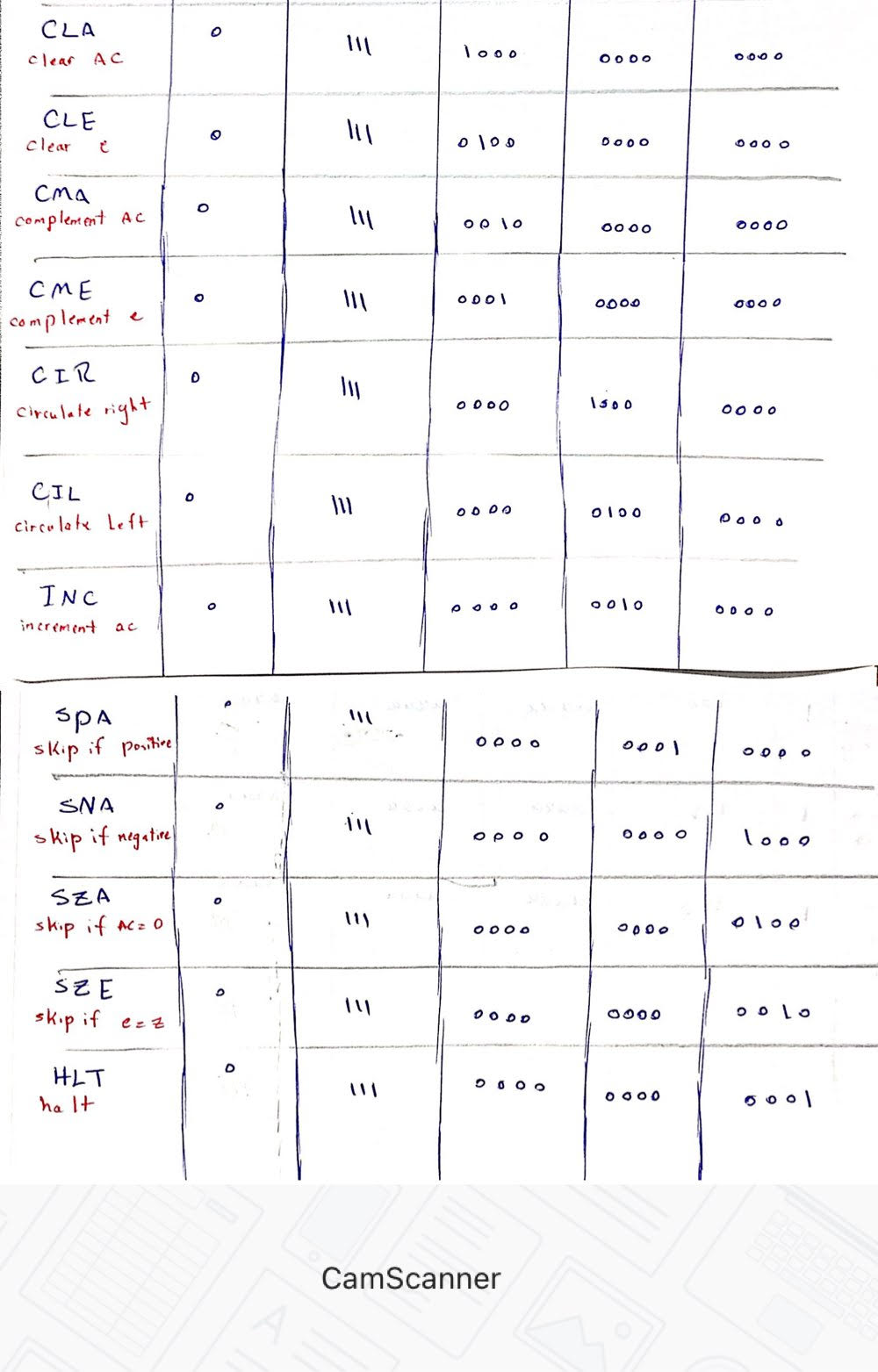
# Instruction set architecture

در این بخش به طراحی قالب درستورالعمل هر دستور می‌پردازیم

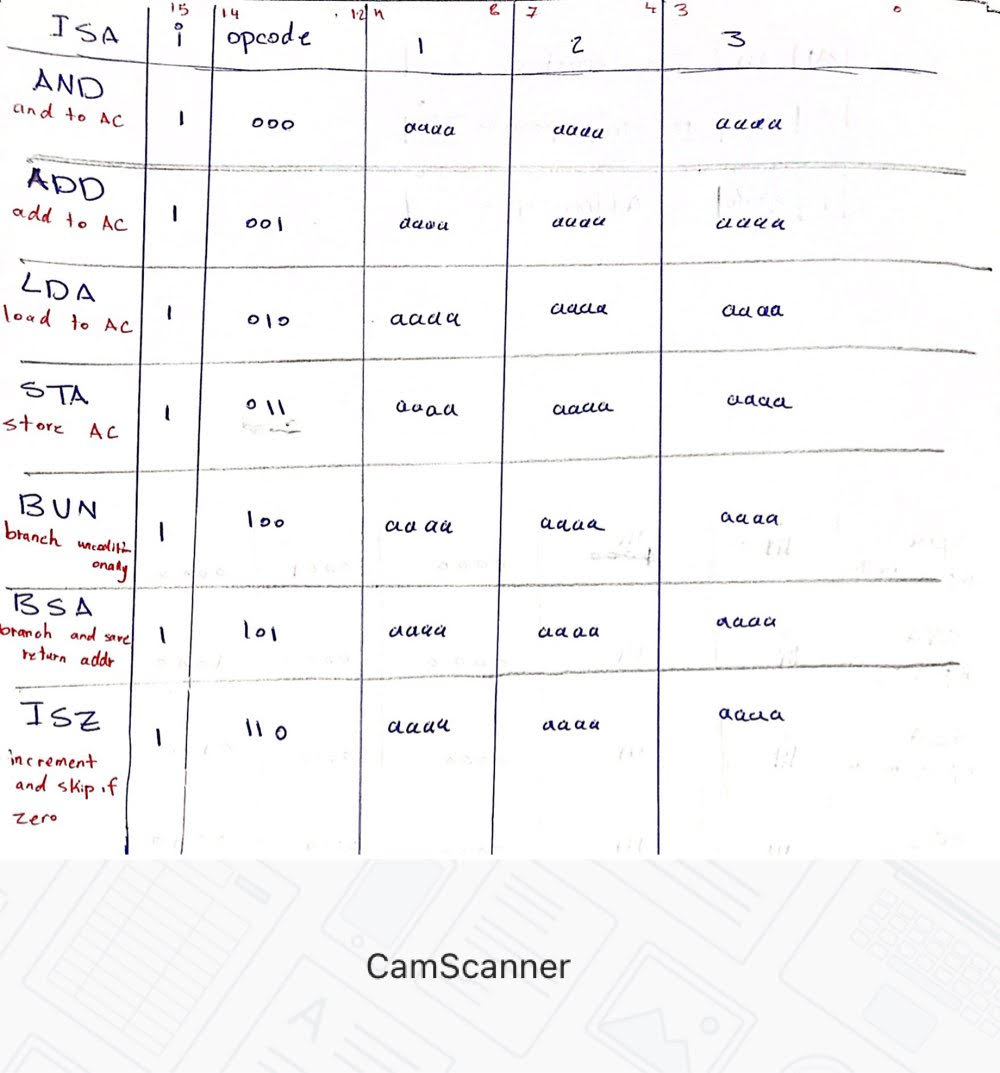
Input-Output instructions



Register-Reference Instructions



Memory-Reference Instructions



در کد برای متمایز کردن هر دستورالعمل از یکدیگر از 2 سیگنال استفاده شده.

1. دیکود شده opcode در d
2. مدل آدرس‌دهی در i

طریقه دیکود شدن opcode در جدول زیر آمده است:

|  |  |
| --- | --- |
| d | opcode |
| 00000001 | 000 |
| 00000010 | 001 |
| 00000100 | 010 |
| 00001000 | 011 |
| 00010000 | 100 |
| 00100000 | 101 |
| 01000000 | 110 |
| 10000000 | 111 |

همچنین برای ریزعملیات (micro operation) ها از زمان T استفاده می‌کنیم تا مشخص باشد هر دستور در چه زمانی باید اجرا شود.

به جدول زیر توجه کنید:

|  |  |
| --- | --- |
| code | T |
| 0000000000000001 | 0 |
| 0000000000000010 | 1 |
| 0000000000000100 | 2 |
| 0000000000001000 | 3 |
| 0000000000010000 | 4 |
| 0000000000100000 | 5 |
| 0000000001000000 | 6 |
| 0000000010000000 | 7 |
| 0000000100000000 | 8 |
| 0000001000000000 | 9 |
| 0000010000000000 | 10 |
| 0000100000000000 | 11 |
| 0001000000000000 | 12 |
| 0010000000000000 | 13 |
| 0100000000000000 | 14 |
| 1000000000000000 | 15 |

در کد داده شده ابتدا به کمک PC می‌فهمیم چه دستوری باید اجرا شود. بعد مختویات دستور را از حافظه اصلی خوانده و داخل IR می‌ریزیم.

در مراحل بعدی بر اساس opcode, i می‌فهمیم کدام دستور باید اجرا شود و با کمک T در زمان‌های مختلف، ریزعملیات‌های مربوط به آن را اجرا می‌کنیم.

همچنین در کد داده شده برای اینکه بفهمیم در هر مرحله چه دستوری را انجام دهیم و آن دستور مربوط به کدام دستورالعمل است از d, T, i استفاده می‌کنیم.

برای دستورات I/O و Register-Reference از مابقی IRبرای مشخص کدرن دستور استفاده می‌کنیم.

برای دستورات حافظه‌ای هم 12 بیت کم ارزش IR، خانه خافظه مورد نظر ما در حافظه اصلی است.