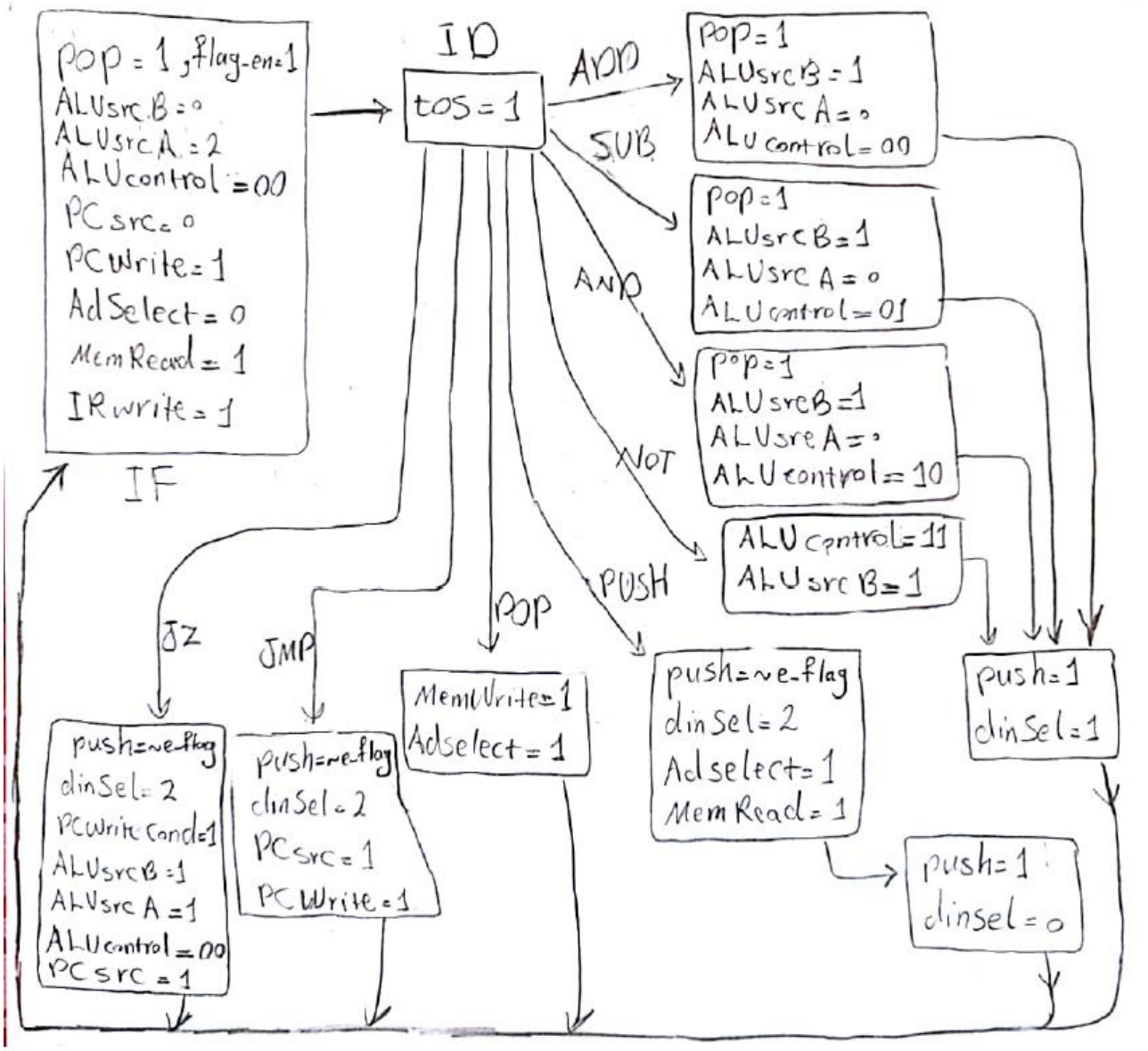
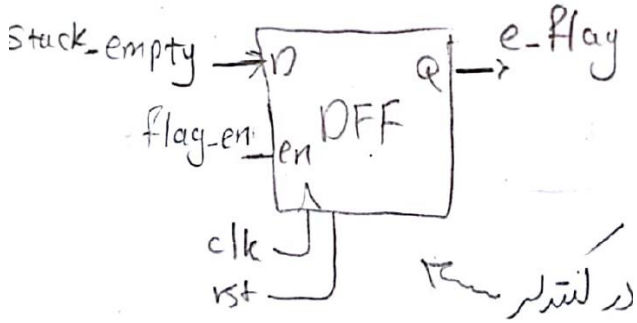
کنترلر :



ALUcontrol	function
00	Out ← A+B
01	Out ← A-B
10	out ← A AND B
11	out ← ~B



توضیح: همان طور که در استیت های IF و ID در کنترلر، مشخص است، زمانی که هنوز نوع دستور مشخص نشده سیگنال های pop, tos فعال می شوند. علت این کار این است که از آنجا که در هر دستور از ۸ دستور ممکن پردازنده، ما نیاز به حداقل یک آپرند که در سر استک قرار دارد، داریم، پس به عنوان پیش فرض و سریعتر کردن پردازنده در اکثر دستورات (۷ دستور از ۸ دستور) این کار انجام می شود. ابتدا در استیت IF سیگنال pop فعال شده تا پس از خارج شدن از این استیت، عنصر پاپ شده روی d_out استک قرار گیرد. سپس در استیت ID، سیگنال tos فعال شده است. در نتیجه، هنگام خارج شدن از این استیت مقدار قبلی d_out (عنصر pop شده) در رجیستر popped data ذخیره شده و d_out به عنصر سر استک (عنصر دوم از بالا زمانی که در استیت IF قرار داشتیم) آپدیت شود. در نتیجه زمانی که از ID خارج شدیم و در استیت سوم قرار داریم، آپرند اول روی d_out (پاپ نشده) و آپرند دوم (عنصری که در ابتدا روی سر استک بود) در رجیستر popped data است (پاپ شده). این کار به انجام دستورات ADD, SUB, AND (از هر دو آپرند استفاده می کنند) و NOT, POP (از عنصر موجود در popped data استفاده می کنند) سرعت می بخشد. در سه دستور PUSH, JMP, JZ که نیاز به POP کردن نبوده است، در استیت سوم متصدی این دستورات با فعال کردن سیگنال های push, clrSel=2، دیتای پاپ نشده دوباره پوش می شود (قبل از اضافه شدن احتمالی دیتای جدید در استک). این پوش کردن دیتای پاپ نشده (در این سه دستور) سربار اضافی cycle ایجاد نمی کند. چون برای انجام این دستورات، سوا از عملیات پوش دیتای پاپ نشده، به حداقل یک cycle دیگر، غیر از IF, ID نیاز است.

* چنانچه زمانی که استک فاقد داده است و خالی است، یکی از ورودی های pop یا tos فعال شود، یک مقدار پیشفرض از استک ظاهر می شود.
همچنین در استیت IF که pop صورت می گیرد، وضعیت پرچم empty در کنترلر ضبط می شود و در سیکل سوم دستورات JMP و JZ و PUSH از این مقدار استفاده می شود که اگر در ابتدا استک تهی است، در این سیکل وارد استک نشود.