



Основи рачунарске технике 2

Посматра се део рачунара који чине меморија и процесор.

Меморија је капацитета 2^{16} бајтова. Ширина меморијске речи је 1 бајт.

Процесор је са једноадресним форматом инструкција. Подаци су целобројне величине без знака. Подаци и адресе су дужине 16 бита и заузимају по две суседне меморијске локације, при чему се старији бајт налази на нижој, а млађи бајт на вишој адреси. Инструкције су дужине 1, 2 или 3 бајта.

У процесору постоји програмски бројач PC дужине 2 бајта, адресни регистар меморије MAR дужине 2 бајта, прихватни регистар податка меморије MDR дужине 1 бајт, прихватни регистар инструкције IR дужине 3 бајта, акумулатор ACC и прихватни регистар податка B оба дужине 2 бајта, регистри опште намене R0 до R7 дужине 2 бајта, програмска статусна реч PSW дужине 1 бајт, указивач на врх стека SP дужине 2 бајта и указивач на табелу са адресама прекидних рутина IVTP дужине 2 бајта. Битови 0 до 4 PSW регистра одговарају индикаторима I, N, C, Z и V, респективно.

Бит 7 првог бајта инструкције има вредност 1 за безадресне инструкције и инструкције скока, док бит 6 првог бајта инструкције има вредност 0 за безадресне инструкције и вредност 1 за инструкције скока.

Безадресне инструкције су инструкције повратка из потпрограма (RTS), повратка из прекидне рутине (RTI), инструкција логичког померања акумулатора удесно за једно место (LSR), инструкција преноса акумулатора на врх стека (PUSH), инструкција преноса са врха стека у акумулатор (POP). Дужина инструкција је 1 бајт.

Инструкције скока су инструкција условног скока уколико је резултат нула (BZ), безусловног скока (JMP) и скока на потпрограма (JSR). Инструкција BZ се реализује као релативни скок у односу на текућу вредност програмског бројача PC, а померај је 8 битна целобројна величина са знаком дата другим бајтом инструкције. Дужина инструкција је 2 бајта. Инструкције JMP и JSR се реализују као апсолутни скокови, а адреса скока је дата 2 и 3 бајтом инструкције, при чему је старији бајт адресе скока дат другим, а млађи бајт трећим бајтом. Дужина инструкција је 3 бајта.

Битовима 5 до 0 првог бајта инструкција специфицира се код операције за безадресне инструкције и инструкције скока. На основу тога су за инструкције RTS, RTI, PUSH, POP, LSR, BZ, JMP и JSR усвојени кодови операција 000000, 000001, 000010, 000011, 000100, 000101, 000110 и 000111, респективно.

Бит 7 првог бајта инструкције има вредност 0 за адресне инструкције. Адресне инструкције су инструкције преноса у акумулатор (LOAD) и из акумулатора (STORE), аритметичка инструкција сабирања (ADD), инструкција инкрементирања аргумента без утицаја на акумулатор (INC). Дужина инструкције је 1, 2 или 3 бајта и зависи од специфицираног начина адресирања. Битовима 6 до 4 првог бајта инструкција специфицира се код операције. На основу тога су за инструкције LOAD, STORE, ADD и INC усвојени кодови операција 000, 001, 010 и 011, респективно. Битовима 3 до 0 дефинише се начин адресирања и регистар опште намене уколико се користи у задатом начину адресирања. Бит 3 вредношћу 1 специфицира регистарско директно адресирање, а битовима 2 до 0 један од регистара опште намене R0 до R7. Дужина инструкције је 1 бајт. Бит 3 вредношћу 0 одређује да се битовима 2 до 0 специфицирају следећа адресирања: 000-меморијско директно адресирање, 001-меморијско индиректно адресирање, 010-непосредно адресирање и 011-PC релативно адресирање. Код меморијског директног адресирања и меморијског индиректног адресирања други и трећи бајт инструкције садрже адресу меморијске локације, при чему је старији бајт адресе меморијске локације дат другим, а млађи бајт трећим бајтом. Код меморијског индиректног адресирања, адреса дужине 16 бита заузима две суседне меморијске локације, при чему се старији бајт налази на нижој, а млађи бајт на вишој адреси. Код непосредног адресирања други и трећи бајт инструкције садржи 16 битни податак, при чему се старији бајт налази на нижој а млађи бајт на вишој адреси. Дужина инструкције за ова три адресирања је 3 бајта. Код PC релативног адресирања

други бајт инструкције садржи 8 битни померај који је дат као целобројна величина са знаком. Дужина инструкција је 2 бајта.

У случају грешке при адресирању у инструкцији, инструкција треба да буде без дејства (прелази се на извршавање следеће инструкције).

Стек расте према вишим меморијским локацијама, а регистар SP указује на прву слободну меморијску локацију.

Захтеве за прекид може да генерише осам контролера периферија по линијама *intr7* до *intr0*. Процесор на њих реагује или не реагује у зависности од тога да ли се у разреду PSWI регистра програмске статусне речи $PSW_{7...0}$ налази вредност 1 или 0, респективно. Адресе прекидних рутина налазе се у улазима 0 до 7 табеле са адресама прекидних рутина. Улаз 0 одговора захтеву за прекид по линији *intr7*, улаз 1 одговора захтеву за прекид по линији *intr6*, ..., улаз 7 одговора захтеву за прекид по линији *intr0*. Почетна адреса табеле са адресама прекидних рутина се налази у регистру IVTP дужине 2 бајта.

У оквиру хардверског дела опслуживања захтева за прекид на стек са стављају регистри PC, PSW као и првих пет регистара опште намене (R_0 - R_4), респективно. Након постављених података на стеку, I бит PSW регистра (PSWI) се ресетује (уписује се вредност 0).

K2) (20)

- Написати израз за генерисање сигнал *gradr* који је активан уколико прочитана инструкција има грешку при адресирању.
- Нацртати дијаграм тока фазе читања инструкције, фазе формирања адресе и читања операнда, фазе извршавања инструкције и фазе опслуживања захтева за прекид. Искористи направљен сигнал под а) што је раније могуће.
- Написати изразе за генерисање сигнала свих начина адресирања и дужина инструкција.
- Написати изразе за генерисање сигнала операција и то само за PUSH и LOAD.

K3) (20) Блок *intr* креће са фазом опслуживања прекида уколико се у флип-флопу INTR налази вредност 1. Након тога се врши провера да ли је у току извршавања претходне три фазе текуће инструкције стигао захтев за прекид по некој од линија *intr7* до *intr0* и да ли се у разреду PSWI регистра $PSW_{7...0}$ налази вредност 1. Уколико није, завршава се са фазом опслуживања прекида у блоку *intr* и прелази се на фазу читања следеће инструкције у блоку *fetch*. Уколико јесте, наставља се са опслуживањем прекида. По завршетку фазе опслуживања прекида, уписује се вредности 1 у флип-флоп FETCH, чиме се стартује блок *fetch* и креће са фазом читања следеће инструкције, док се уписивањем вредности 0 у флип-флоп INTR зауставља блок *intr*.

- Нацртати структуру операционе јединице блока *intr* процесора. **Приказати компоненте релевантне само за овај блок.**
- Дати секвенцу управљачких сигнала (по корацима - *step*) операционе и управљачке јединице у случају да се користи микропрограмска реализација управљачке јединице. Навести изразе свих сигнала услова који су коришћени у секвенци.
- Дати формат микроинструкције у коме су кодирани сигнали операционе и управљачке јединице, услови скокова и адресе скока. **Попуњавати искључиво приложене таблице датог формулара.**
- Нацртати структуру управљачке јединице микропрограмске реализације и приказати како се генеришу сигнали операционе и управљачке јединице на основу формата микроинструкције.
- Приказати садржај микропрограмске меморије за прва три корака.

Напомене: На испиту нису дозвољена никаква помоћна средства, ни калкулатори, ни литература. Испит траје 3 сата. Није дозвољен излазак првих сат времена. **Обавезно назначити на формулару да ли се поправља K2.** Студент је дужан да пише читко и шеме да црта прегледно. При цртању шема, обавезно назначити ширину сигнала. Обавезна је предаја формулара и вежбанке.



Електротехнички факултет у Београду
Катедра за рачунарску технику и информатику

07.02.2017.

Основи рачунарске технике 2

Презиме и име	Индекс (гггг/бббб)	Желим да поправим К2:
		ДА НЕ

Колоквијум 3 - формулар

Таблица - формат микроинструкције

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31

Таблица - безусловни скокови

Назив сигнала	сс [hex]

Таблица - условни скокови

Назив сигнала	сс [hex]	Услов скока

Таблица - вишеусловни скокови

Назив сигнала	сс [hex]	Скок (формат: услов скока → адреса)