Prozessortechnik

Aufgabe 1 - Prozessor mit Akku-Architektur

Folgende Abbildung zeigt einen Mikrocontroller mit Akku-Architektur.



Frage 1.1 (8 Punkte): Erläutern Sie stichwortartig die einzelnen Komponenten im Blockschalt-bild und ihre Funktion (Programmspeicher, Datenspeicher, Steuerwerk und Rechenwerk mit Details)

Frage 1.2 (4 Punkte): Addieren und Subtrahieren: Erläutern Sie den Ablauf der Befehle „ADD Addr“ und „SUB Addr“ in Stichworten und unter Verwendung folgender Abbildungen.





Hinweis: Startpunkt ist der dekodierte Befehl am Ausgang des Dekoders. Zeichnen Sie die jeweils aktiven Signale (Leitungen) im Diagramm nach.

Frage 1.3 (6 Punkte): Folgender Ausschnitt aus dem Steuerwerk des Prozessors beschreibt die Ausführung eines Befehls. Erläutern Sie (1) den Aufbau der HDL-Struktur, (2) den Zweck der beiden Zustände, (3) die Aktionen im ersten Takt und zweiten Takt, (4) in welchem Takt der Programmzähler umprogrammiert wird, (5) in welchem Takt der folgende Befehl geladen wird, (6) welcher Befehl als nächstes ausgeführt wird.

...

-- state transitions & actions

logic\_next\_state\_and\_actions: process(current\_state, T\_OPCODE)

begin

case current\_state is

when Z1 => -- one clock cycle per operation

case T\_OPCODE is

...

when JMP => -- stop pipeline, load PC

T\_Pipe\_EN<='0'; T\_PC\_EN<='1';

T\_PC\_LOADA<='1';

next\_state<=Z2; -- 2nd cycle

...

when others => null;

end case; -- opcode one cycle

when Z2 => -- second clock cycle per operation

case T\_OPCODE is

...

when JMP => -- re-animate pipeline

T\_Pipe\_EN<='1'; T\_PC\_EN<='1';

T\_PC\_LOADA<='0';

next\_state<=Z1;

...

when others => null;

end case; -- opcode two cycles

when others => null;

end case; -- states

end process logic\_next\_state\_and\_actions;

...

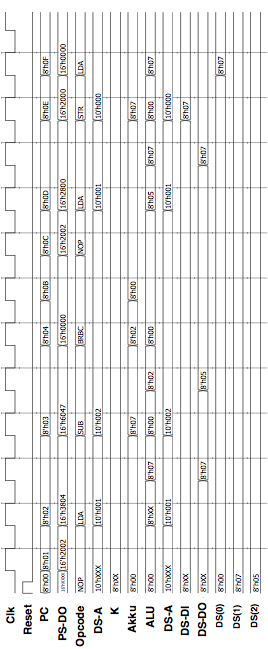
Frage 1.4 (4 Punkte): Skizzieren Sie ein Zustandsdiagramm des oben beschriebenen Automa-ten (Frage 1.3). Ergänzen Sie die Zustandsübergänge für die Befehle „ADD Addr“ und „LDI K“ (load Akku immediate).

Frage 1.5 (4 Punkte): Sprungbefehl: Erläutern Sie den Ablauf des Befehls „JMP Addr“ in Stichworten und unter Verwendung folgender Abbildungen. Welchen Zweck hat es, die Pipeline im ersten Takt anzuhalten? Was geschieht, wenn nicht angehalten wird?





Frage 1.6 (8 Punkte): Das Zeitdiagramm auf der folgenden Seite beschreibt den Ablauf eines Programms. (1) Erläutern Sie den grundsätzlichen Ablauf, (2) rekonstruieren Sie den Programmtext aus dem Ablauf (bitte mit Zeilennummern), (3) erläutern Sie den Ablauf des Befehls SUB Addr im Detail (Inhalt Akku und Speicher für Operanden und Ergebnis), (4) Erläutern Sie den Ablauf des Befehls BRBC im Detail (Programmzähler, folgender Befehl, Ursprung des Befehls NOP im Anschluss an BRBC).





Aufgabe 2 - Prozessor mit Register-Architektur

Anstelle des Akkus wird eine Registerbank mit 16 Registern eingeführt. Jedes Register kann als Akku verwendet werden. Folgendes Blockschaltbild beschreibt die Realisierung.



Frage 2.1 (6 Punkte): Beschreiben Sie die wesentlichen Unterschiede zur Akku-Architektur aus Aufgabe 1 (Laden von Registern, ALU-Operationen, Laden und Speichern von Operan-den im Datenspeicher).

Frage 2.2 (6 Punkte): Nennen Sie Unterschiede im Befehlssatz und in der Befehlskodierung (Beispiele: Sprungbefehle, Laden und Speichern von Ergebnissen, ALU-Operationen).

Frage 2.3 (4 Punkte): Gibt es Vorteile, die die Registerarchitektur gegenüber der Akku-Architektur bringt (Geschwindigkeit, Programmierbarkeit)?

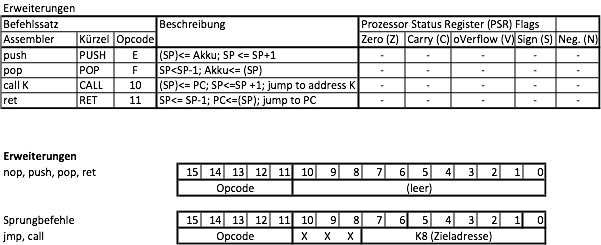
Frage 2.4 (6 Punkte): Schreiben Sie das rekonstruierte Programm aus Aufgabe 1.6 auf die Registerarchitektur um. Welche wesentlichen Unterschiede gibt es?

Aufgabe 3 - Erweiterungen des Prozessors (Akku-Architektur)

Zur Unterstützung der strukturierten Programmierung durch Unterprogramme soll der Befehlssatz des Mikrocontrollers durch folgende Befehle erweitert werden:

* CALL Addr: Ruft Unterprogramm auf. Dieser Befehl funktioniert wie ein Sprung-befehl, sichert jedoch zusätzlich den alten Stand des Befehlszählers im Daten-speicher, damit dieser für den Rücksprung verwendet werden kann.
* RET: Rückkehr aus dem Unterprogramm. Als Rücksprungadresse verwendet das Unterprogramm den im Datenspeicher abgelegten Programmzähler.
* PUSH: Sichert den Inhalt des Akkus im Datenspeicher. Auf diese Weise wird vermieden, dass Zwischenergebnisse im Akku beim Sprung ins Unterprogramm überschrieben werden.
* POP: Stellt den Inhalt des Akkus aus dem Datenspeicher wieder her.

Die Sicherung des Programmzählers und Akkus im Datenspeicher geschieht in einem gesonderten Bereich (Stapel, engl. Stack), der mit Hilfe eines Stapelzeigers (SP für engl. Stack Pointer) adressiert wird. Folgende Übersicht zeigt die Befehlserweiterung.



Frage 3.1 (6 Punkte): Erläutern Sie die Funktion der Befehle im Detail (Akku, Stapel, PC).

Frage 3.2 (4 Punkte): Welche Befehle werden vom Hauptprogramm aufgerufen? Welche Befehle werden vom Unterprogramm aufgerufen? Skizzieren Sie den Ablauf des Aufrufs eines Unterprogramms in einem Aktivitätsdiagramm.

Frage 3.3 (6 Punkte): Zur Realisierung o.g. Befehle muss der Prozessor erweitert werden. Ergänzen Sie die erforderlichen Erweiterungen im Blockschaltbild. Hinweis: Verwenden Sie folgende Vorlage. Behandeln Sie Akku und Programmzähler getrennt.



Frage 3.4 (4 Punkte): Wie viele Takte benötigen die Befehl jeweils: CALL Addr, RET, PUSH, POP. Begründen Sie Ihre Antwort.

Frage 3.5 (4 Punkte): Folgendes Diagramm beschreibt den Stapel im Datenspeicher mit dem SP als Zeiger nach Ablauf eines Befehls. Ergänzen Sie das Diagramm für folgende Schritte: (0) Startpunkt, (1) CALL Addr, (2) PUSH, (3) POP, (4) RET. Hinweis: Verwenden Sie das Diagramm auf der folgenden Seite als Vorlage. Gefragt sind Inhalt des Stapels und Position des Stapelzeigers.



Frage 3.6 (8 Punkte): Das Zeitdiagramm auf der folgenden Seite beschreibt den Ablauf eines Programms. (1) Erläutern Sie den grundsätzlichen Ablauf, (2) rekonstruieren Sie den Programmtext aus dem Ablauf (bitte mit Zeilennummern), (3) erläutern Sie den Ablauf der Befehle PUSH und POP im Detail (Inhalt Akku und Stapel), (4) Erläutern Sie den Ablauf der Befehle CALL und RET im Detail (Programmzähler, folgender Befehl, Speicherung und Restaurierung des Programmzählers, Rücksprung an welche Adresse, Ursprung und Zweck des Befehls NOP im Anschluss an CALL).

